NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

TEMPO SOFTWARE MODIFICATIONS FOR SEVER EVALUATION

by

Ronald F. Clemens

September 2009

Thesis Advisor:	Gary O. Langford
Second Reader:	Rodney W. Johnson

REPORT DOCUMENTATION PAGE			*Form Approved OMB No. 0704-0188*
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.			
1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** September 2009	**3. REPORT TYPE AND DATES COVERED** Master's Thesis	
4. TITLE AND SUBTITLE TEMPO Software Modification for SEVER Evaluation			**5. FUNDING NUMBERS**
6. AUTHOR(S) Ronald F. Clemens			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000			**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A			**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. government.			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited			**12b. DISTRIBUTION CODE** A

13. ABSTRACT (maximum 200 words)

In this thesis, we present a software package update facilitating the evaluation of the Systems Engineering Value Equation with Risk (SEVER) using the TEMPO Military Planning Game originally developed by General Electric in the 1960's. Currently it is used by Naval Postgraduate School to train future military decision makers in making critical resource allocation decisions. The intent of this development facilitates creating a software decision tool adaptable to different software-based resource allocation models such that decision makers are presented with customized, relevant information for both 'satisfaction' and 'optimization' decisions along with a probability of successful outcome prior to executing the decision and observing the effects. The software package development (identified as TEMPO Version 3) involved porting software code from C++ into Visual Basic.NET, integrating MATLAB code for numerical strategy execution, and allowing for future strategy generation and statistical analysis to compare against SEVER-predicted outcomes. SEVER is the algorithm proposed by Langford and Hyuhn (2006) that facilitates Systems Engineering decision making. TEMPO Version 3 is the particular software application proposed to evaluate the SEVER algorithm once it is developed into a software package. SEVER was not implemented; however, it was considered during the development of the updated software package.

14. SUBJECT TERMS Decision, Decision Analysis, Decision Process, System Engineering Tool, SEVER, resource allocation, military planning, software tool, strategy evaluation			**15. NUMBER OF PAGES** 131
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UU

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89) Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

TEMPO SOFTWARE MODIFICATIONS FOR SEVER EVALUATION

Ronald F. Clemens
Systems Engineer, Naval Undersea Warfare Center, Division Newport, VA
BSME, Cornell University, 2003
M.Eng, Cornell University, 2004

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN SYSTEMS ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
September 2009

Author:	Ronald F. Clemens
Approved by:	Gary O. Langford Thesis Advisor
	Rodney W. Johnson Second Reader
	David Olwell, PhD Chairman, Department of Systems Engineering

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

In this thesis, we present a software package update facilitating the evaluation of the Systems Engineering Value Equation with Risk (SEVER) using the TEMPO Military Planning Game originally developed by General Electric in the 1960s. Currently, it is used by Naval Postgraduate School to train future military decision makers in making critical resource allocation decisions. The intent of this development facilitates creating a software decision tool adaptable to different software-based resource allocation models such that decision makers are presented with customized, relevant information for both 'satisfaction' and 'optimization' decisions along with a probability of successful outcome *prior to* executing the decision and observing the effects. The software package development (identified as TEMPO Version 3) involved porting software code from C++ into Visual Basic.NET, integrating MATLAB code for numerical strategy execution, and allowing for future strategy generation and statistical analysis to compare against SEVER-predicted outcomes. SEVER is the algorithm proposed by Langford and Hyuhn (2006) that facilitates Systems Engineering decision making. TEMPO Version 3 is the particular software application proposed to evaluate the SEVER algorithm once it is developed into a software package. SEVER was not implemented; however, it was considered during the development of the updated software package.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

It is difficult for me to express how appreciative I am for the support and encouragement I have received from my family, friends, and colleagues throughout the past few years. First, I have to thank Joseph Ho for his software expertise in Visual Basic.NET and C++. Without his technical support and positive conversation, the effort presented within this thesis would have, at best, been a far-fetched dream.

I would also like to thank all my colleagues at the Naval Undersea Warfare, Division Newport—most importantly, my supervisor, Robert Chaves—for their understanding and acceptance while I worked on this thesis alongside my daily tasking. (And thank you for the funding, too!) I admit my patience wore thin at times from "burning the candle at both ends," but your encouraging thoughts and words soothed things on a daily basis.

My advisors, Gary Langford and Rod Johnson deserve special thanks for the many teleconference sessions they held with me over the past few years. These sessions provided most of the technical content for me to understand SEVER and update the software successfully. Their support was critical in executing this work and keeping me on the right track. A special thanks to Gary for walking me through discussions on the details of SEVER for hours at a time. Your expertise and willingness to discuss this certainly did not go unnoticed.

I would also like to thank my mother, Judilyn; my stepfather, Steve; my sister, Amilyn; and my grandparents, Jim and Judy. Without all your support and faith in me, I would not have the frame of mind to succeed today.

Lastly, I have to thank my wife, Monica, for sacrificing so much of her time so that I could perform this work. Without her support in just being here for me, my perspective on the life/work balance would have been lost.

Thank you everybody!

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. MOTIVATION

In light of today’s fast-paced and dynamic global environment, it is imperative for United States Department of Defense (DoD) decision makers to evaluate and execute critical strategic decisions efficiently. Among the most important of these decisions are managing current assets, funding development of promising future technologies, converting state-of-the-art technologies into war-fighting applications, and inserting Commercial-off-the-Shelf (COTS) technology into military systems. Understanding the factors involved in making these decisions can be complex and convoluted. Simplifying and clarifying the interactions between these factors would benefit decision makers at every level. Decision support systems exist within the DoD to guide the decision maker in making a proper decision.

According to the Defense Acquisition Guidebook published in 2003 (updated in 2008), the DoD has three principal decision support systems in place. These are the Planning, Programming, Budgeting and Execution (PPBE) Process, Joint Capability Integration and Development System (JCIDS), and the Defense Acquisition System. The PPBE process is used to craft plans and programs that satisfy the demands of the National Security Strategy within resource constraints. The JCIDS provides a systematic method established by the Joint Chiefs of Staff for assessing gaps in military joint warfighting capabilities and recommending solutions to resolve these gaps. The Defense Acquisition System is a management process by which the DoD acquires weapon systems and automated information systems. Together, the three systems provide an integrated approach to strategic planning, identification of needs for military capabilities, systems acquisition, and program and budget development.

Decisions are frequently outcomes of trade studies. A trade study, or trade-off study, is the activity of a multidisciplinary team to identify the most balanced technical solutions among a set of proposed viable solutions (Federal Aviation Administration [FAA], 2006). Identifying ‘the most balanced technical solution’ involves clearly understanding the set of multiple, competing criteria and subsequently ranking their importance. For the purposes of this research, decisions and trade studies are considered synonymous. Both decisions and trade studies involve evaluating the pros and cons, or risks and rewards of possible outcomes. If all options constituting a decision's outcome could be analyzed thoroughly (e.g., through some inductive or deductive process), such that the risks and rewards for each were objectively defined and understood, we surmise that the possible outcomes of a decision could be evaluated objectively. And, just as importantly, if this risk/reward analysis could be automated for a given situation, perhaps the time to make that decision could be reduced. Then the focus would center on increasing a decision maker's effectiveness in evaluating the possible outcomes. By assessing the likelihood and consequences of outcomes and decreasing the time to make a decision, the risks of making the wrong decision at the wrong time could be diminished.

However, if information on the interacting factors that confound a decision is incomplete or unorganized, the limited time in which to make a decision could be wasted through physically and mentally organizing the information. Disorganization could result in inaccurate assessment of an outcome's consequence. Coupling today's powerful computational processing with a new, innovative methodology based on a total systems perspective to organizing information, the author posited that a decision evaluation tool which bridges the gap between 'satisfaction decisions' and 'optimization decisions' could be developed to help increase performance and quality of a decision maker's execution.

The research and development presented in this thesis developed the groundwork for incorporating and testing an algorithm that is the core of such a software tool. This algorithm is currently known as the Systems Engineering Value Equation with Risk (SEVER), and was originally derived by Gary Langford in 2006.

B. BACKGROUND

"Decision" is a common term used in the English language covering a variety of topics. In sports, a decision means a win or loss. In law, a decision determines a person's guilt or innocence. Regarding this thesis, a decision is a position or opinion or judgment reached after consideration (American Heritage Dictionary, 2004). Every day, people must make decisions. The impacts of these decisions directly reflect the amount of time required to spend considering outcomes prior to making that decision, or deciding. There are many areas of research relating to analyzing and forecasting a decision's potential impact or outcome. If the decision warrants the effort, decision theory can be applied. Resnik (1987) states:

> Decision theory is the product of the joint efforts of economists, mathematicians, philosophers, social scientists, and statisticians toward making sense of how individuals and groups make or should make decisions. The applications of decision theory range from very abstract speculations by philosophers about ideal rational agents to practical advice from decision analysts trained in business schools.…It is thus usual to divide decision theory into two main branches: normative (or prescriptive) decision theory and descriptive decision theory. Descriptive decision theorists seek to find out how decisions *are* made—they investigate ordinary mortals; their colleagues in normative decision theory are supposed to prescribe how decisions ought to be made—they study ideally rational agents.

This thesis focused on normative decision theory. Further, two classes of normative decisions were considered: 'satisfaction' and 'optimization' decisions. Let a satisfaction decision be defined as achieving the highest expected utility under a given situation. According to Bernoulli (1738), the determination of the *value* of an item must not be based on its *price*, but rather on the *utility* it yields. The price of the item is dependent only on the thing itself and is equal for everyone; the utility, however, is dependent on the particular circumstances of the person making the estimate. Thus, the best outcome of a satisfaction decision is one that reflects the highest expected utility.

In the other decision type, 'optimization,' decisions are based on optimization theory. According to Jongen et al. (2004), optimization theory is the mathematical study of problems that ask for minimal or maximal values of an objective function on a given domain. Considering this definition, let an optimization decision be defined as finding the minimum or maximum values (i.e., maximizing or minimizing) for a pre-defined, representative mathematical model of the situation (i.e., the objective function). Further, the subject of multiple criteria optimization is the selection of good decisions from a set of alternatives with respect to multiple criteria or objective functions (Steuer, 1986).

These types of decisions usually revolve around conflicting objective functions in which Pareto-efficient solutions exist. In general, Pareto efficiency, or Pareto optimality, is easily described with respect to a game: An outcome of a game is Pareto efficient if there is no other outcome that makes every player at least as well off and at least one player strictly better off (Gametheory.net, n.d.). In the case of multiple criteria optimization, the solution is Pareto efficient if there is no solution for which a particular criterion's solution can be improved without simultaneously degrading another. At the point of Pareto-efficiency, the decision maker's discretion is required to make the ultimate decision, as the existing Pareto-solutions are independently optimal against the conflicting objective functions. The methods used to solve multiple objective optimization problems include Markov Decision Processes (Bellman, 1957) through dynamic programming and reinforcement learning (Sutton & Barto, 1998), multi-attribute utility theory (MAUT) (Keeney & Raiffa, 1976), and evolutionary algorithms (Abraham, Jain, & Goldberg, 2005), to name a few. Each of these methods requires the situation to be represented in the form of a mathematical relationship, or objective equation, in order to execute these numerical methods. A quick summary of each follows.

According to Howard (1960):

> A Markov Process is a mathematical model that is useful in the study of complex systems. The basic concepts of the Markov process are those of "state" of a system and state "transition."…A graphic example of a Markov process is presented by a frog in a lily pond. As time goes by, the frog jumps from one lily pad to another according to his whim of the moment. The state of the system is the number of the pad currently occupied by the frog; the state transition is of course his leap. ...If we focus our attention on the state transitions of the system and merely index the transitions in time, then we may profitably think of the system as a discrete-time process….To study the discrete-time process, we must specify the probabilistic nature of the state transition.

This method is difficult to implement without accurate knowledge of the probabilistic nature state transitions. Many decisions are not consistently encountered, or have previously been statistically modeled.

According to Sutton and Barto (1998), reinforcement learning is learning what to do—how to map situations to actions—so as to maximize a numerical reward signal. The learner is not told which actions to take, as in most forms of machine learning, but instead must discover which actions yield the most reward by trying them.

MAUT is a special case of the Unidimensional Utility Theory involving more than one utility function (Keeney & Raiffa, 1976). Keeney and Raiffa (1976) summarize Unidimensional Utility Theory as follows:

> A decision maker must choose among several alternatives…each of which will eventually result in a consequence describable in terms of a single attribute. The decision maker does not know exactly what consequence will result from each of the various alternatives, but he can assign probabilities to the various possibilities that might result from any course of action.…If an appropriate utility is assigned to each possible consequence and the expected utility of each alternative is calculated, then the best course of action is the alternative with the highest expected utility.

The difference between unidimensional utility theory and MAUT is contained within the number of attributes against which the decision maker must evaluate a decision's outcome. MAUT usually involves multiple optimum solutions, since in most cases the optimization of each individual attribute does not lead to the optimization of all considered attributes. Performing trade studies and considering personal preferences are necessary to evaluate which decision is best once the Pareto-efficient solutions are computed.

Krantz and Kunreuther (2007) developed a subjective expected multi-attribute utility theory (SEMAUT), based on the determination that goals are deemed stable, and therefore the tradeoffs among the different goals can be approximately represented by a multi-attribute utility function. In the case of TEMPO, two constructs distinguish it from the commonly held notion that MAUT has not been associated with prescriptive approaches

1. The notion that preferences are revealed rather than constructed is an important distinction. SEMAUT's frame is in sharp contrast to the idea that preferences are constructed rather than revealed (Tversky, Sattath & Slovic (1988); Tversky, Slovic & Kahneman (1990); Chapman & Johnson

(1995); Slovic (1995)). In the prescriptive paradigm, one concentrates on improving the choice process for a particular context given an understanding of how the computer behaves. Prescriptions should be formed on rules or norms which are constructed rather than designed through preference. SEMAUT focuses on values or utilities and decision weights rather than subjective probabilities, and

2. The notion that there is a multitude of frameworks underscores prescriptive thinking. One must consider the correct temporal framing, how to commit resources to meet a goal, how should the decision-weighting process be enacted, and how to set the goals correctly in the context of the previous play.

Evolutionary algorithms are based on the following three methods: genetic algorithms, evolution strategies, and evolutionary programming (Baech et al., 2000). In general, Baech et al. (2000) explain evolutionary algorithms as:

> [Those that] utilize the collective learning process of a population of individuals.[1] Usually, each individual represents a search point in the space of potential solutions to a given problem. Additionally, individuals may also incorporate further information; for example, strategy parameters of the evolutionary algorithm.

Each of these methods is mentioned as a potential way to solve certain types of problems involving decision outcomes. The intent of this discussion is to provide background information on other methods available, not on the details of their effectiveness or whether one method is preferred.

Regarding ‘satisfaction decisions,’ a proper framework is required to understand the decision at hand in better detail (i.e., the information is presented in a hierarchical fashion to facilitate comparison between the alternatives) so that an educated decision can be made. These methods propose to decompose the decision such that the decision maker has a more objective dataset in order to make the most satisfying choice. Analytic network/hierarchy processes (ANP/AHP), both proposed by Saaty in 2001, and decision trees are two methods that establish a hierarchical view of a particular situation.

[1] The term ‘individual’ refers to a single member containing a chromosome or genome that usually contains at least a representation of a possible solution to the problem being tackled. Other information such as certain strategy parameters and the individual’s fitness value are usually also stored in each individual.

Generally, two perspectives exist in solving problems: “top-down” and “bottom-up” (Lakatos, 1978). According to Blanchard and Fabryky (2006). A top-down approach views the system as a whole, understanding how the system components effectively perform together. In contrast, a bottom-up approach refers to viewing a system by focusing on the lowest level system components and assembling upward. In general, Decision Theory methods such as the Markov Decision Process and MAUT rely on a bottom-up approach. That is, they require a numerical/mathematical model with assigned probabilities of the situation in order to compute the optimal outcome. These methods are concerned with solving an optimization decision problem. In cases based on laws of physics or proven empirical relationships, those methods work because software can be used to perform the intense computations. In these cases, parameter relationships through equations are derived, random inputs are provided, and computations or simulations are performed to provide the absolute range of possible outcomes as a probability distribution. The solution with the highest expected value, or utility, is considered the best. These models account for uncertain conditions by tuning particular parameters to calculate a new cumulative probability distribution for each case. In this sense, lower level model parameters are updated and give rise to a new system-level outcome.

Many other methods, one being ANP/AHP, are more robust because they are typically applied in a top-down manner. The importance of each parameter in a decision varies from person to person. These methods allow the decision maker to provide custom criteria importance values in the form of criteria weighting, which more accurately represents the human behavioral aspect of decision making.

SEVER is an algorithm that is most useful when applied in a top-down manner, yet is built on a bottom-up framework. SEVER is both an algorithm and a mathematical method used to capture and decompose a situation, or system, down to its lowest level elements. In fact, it is a Systems Engineering (SE), top-down, approach that provides a system-level view of the implications of executing a certain decision from the basic elements upward. Because of its SE approach, uncertainty is considered an input to SEVER in the form of *risk*. With respect to ‘satisfaction decisions,’ SEVER provides a decomposition framework in which to view a particular situation. With respect to ‘optimization decisions,’ SEVER provides the mathematical framework for comparing different potential outcomes so the optimal outcome for that situation can be discovered.

One particular application of interest to the DoD involves training future program managers with the skill of allocating resources efficiently. A game has existed since the 1960s to train students at the Naval Postgraduate School in this skill. It was originally developed by General Electric’s TEMPO Think Tank as a military budget allocation game and is known as the TEMPO Military Planning Game. TEMPO is a two-player game consisting of allocating finances on certain available weapon systems. Each weapon system provides a different level of military performance for a defined cost. Each player’s objective is to win a war based on the operating performance of particular inventoried weapons. The game has been modified from its original, paper-based version to a computer command console version to a spreadsheet-based version to its current Microsoft Windows form-based version. Through these evolutions, updates have been made to the game; however, the general intent of the game has remained the same just as has the skill it allows its users to hone: Allocate finances for particular weapon systems in order to win a war against your enemy.

TEMPO was used to implement the constructs to support SEVER. TEMPO was modified to allow real-time feedback on allocation decisions made by the user. Once real-time feedback is integrated, SEVER will be capable of scoring the effectiveness of a decision in real time if the model. Many different strategies can be used during game play; SEVER was the tool used to analyze, or *score*, each strategy.

C. PURPOSE

This document describes the adaptation and update of the TEMPO Military Planning Game, developed by General Electric in the 1960s, to automatically execute a set of rule-based strategies. Further, a trial set of 1000 runs was executed for each strategy. The results of these trial sets are provided. TEMPO was updated to lay the groundwork for the future implementation of SEVER. Also, this work attempted to improve the TEMPO user interface.

By implementing rule-based strategies against the computer's coevolutionary rule-based strategies and recording the outcomes, evaluation of the main tool of the Value Systems Engineering process, SEVER, will be possible (Langford, 2006). For the formative purposes of this thesis, five major strategy types have been designed and implemented: 1. allocate randomly, 2. allocate based on reported probability of war, 3. allocated based on user-defined priority, 4. allocate based on offense or defensive preference, 5. allocate based on intelligence received. Of these, 10 particular (subset) strategies were tested. These strategies are subsets of the strategy types from changing certain parameters. These 10 strategies were executed and their comparative results are described. The methods used to compare these strategies serve as the foundation to evaluate SEVER.

In summary, the work presented in this thesis creates a model on which to perform resource allocation trials based on strategies so that a statistical evaluation of the SEVER algorithm is possible. The TEMPO Military Planning Game (TEMPO) is the model selected to facilitate the evaluation of SEVER. MATLAB was used to create the strategy and analysis capabilities. The entire development is integrated as one software package that is user installed and executed. From this point forward, this software development is referred to as the 'TEMPO Version 3.0.' This software package can continue to be updated to include SEVER, when appropriate.

The particular research question on which the work presented in this thesis is focused is: Can TEMPO be updated and modified to accommodate SEVER? What is required? How can this be achieved?

D. SCOPE

This thesis provides a development description of a decision model capable of both human (manual mode) and computer (automatic mode) execution in order to accommodate SEVER. The model (TEMPO) is designed such that a large number of trials (or 'games') can be executed, and the results recorded, to evaluate the player's performance against the computer. The assimilation of SEVER and its evaluation is planned as a follow-on effort and is not discussed in this write-up.

TEMPO updates provide:

- a mechanism to run a user-defined number of games with various user-selected rule-based strategies
- a means to implement the SEVER evaluation algorithm via a Graphical User Interface (GUI)
- an enhanced the user interface
- a means to graphically present the previous turn's decision results

This thesis also presents the development and performance of 10 particular strategies autonomously played (i.e., computer versus computer) in order to validate TEMPO Version 3. Performance of each strategy is presented statistically.

The scope of the work performed and presented concludes with the ability of TEMPO to record and present performance results of human player and computer automated strategies. This thesis does not present results comparing the actual performance results of these strategies against those predicted by SEVER—only results of the 10 currently coded strategies against each other. The strategy performance results are recorded by automatically executing a select number of trials of a particular user-selected strategy and saved within Extensible Markup Language (XML) log files.

A software development effort was necessary to update the existing TEMPO model from a grid-style interface to a more user-friendly Graphical User Interface (GUI). This development also included preliminary placeholder code so that SEVER can be implemented on the GUI along with special functions (coded in MATLAB) to analyze and present resource allocation results, where appropriate. Special attention was paid to the user interface to facilitate quick learning and reduce the effort required to understand and play the game. Previously, TEMPO was not capable of automatically playing a set number of games. During this thesis work, the software was modified and recoded to collect data for a particular user-selected strategy for a user-specified number of games. The updated software is capable of comparing actual performance of all types of logical rule-based strategies in addition to a player's strategy. Further, a software framework to develop and execute additional strategies in MATLAB was designed. The updated GUI allows human research on executing a strategy under time constraints to be possible, both with and without SEVER.

E. FUTURE WORK

Plans for future work include updating the preliminary placeholder code with the actual SEVER engine and applying the SEVER implementation to other numerical-based models. Particular future work for TEMPO and SEVER implementation will build upon the current strategies with new rule-based strategies, create a graphical user interface to streamline the post-processing and analysis of game log files, and perform a formal statistical analysis using the MATLAB Statistical Toolbox functions.

Future work regarding other models useful for SEVER could be a project schedule modeled in Microsoft Project© software where both financial and temporal decisions are required to successfully complete a project within given management time/budget constraints. The model would be of the software schedule created in Microsoft Project, and exported in an XML file format. Inputs to the SEVER algorithm include decisions regarding the tasks to perform versus the budget and time available. The output of the algorithm would be the risk, as a percentage, that scheduled project would not be successfully completed within the estimated timeframe and budget.

F. DOCUMENT ORGANIZATION

Chapter I contains the motivation and background necessary to understand the work presented within the thesis.

Chapter II contains the background on both TEMPO and SEVER necessary for the reader to understand the subsequent software development effort.

Chapter III contains two subsections. The first section presents the planned software development in order to answer the research question presented. The second section presents the statistical method used to evaluate the performances of each strategy.

Chapter IV describes the software package development, the methods involved in the strategy performance verification, and results of implementing each strategy into TEMPO. Further, it summarizes the software package development and draws conclusions from development of TEMPO's updates and the rule-based strategies to answer the proposed research questions in Chapter I. Also, future development work and research based on the limitations within this thesis are suggested.

The appendices provide more detailed information about TEMPO, SEVER, methods to integrate them, a description of the developed code (i.e., Visual Basic.NET and MATLAB), software installation procedures, and game play instructions and features. A CD is packaged with this publication and contains all the software required to reproduce these results.

II. MODEL DESCRIPTIONS

In order to understand the purpose and results presented within this thesis, the reader requires a sufficient understanding of the model and algorithm used. This chapter is broken into two major sections. The first section focuses on the description and game-play rules for TEMPO. The second section provides an overview of the SEVER algorithm and the method of applying this algorithm to TEMPO.

A. TEMPO MILITARY PLANNING GAME

According to Johnson et al.:

> In the early 1960s, the Department of Defense created a management system, the Planning, Programming, and Budgeting System (PPBS) of considerable complexity to rationalize its resource allocation problems (during a short range timeframe (1–5 years). A major training program was instituted to teach the PPBS and a "game" was created by General Electric's "TEMPO Think Tank" to train people in the use of the new system. (2005)

This training program comes in the form of a symmetric two-sided game. Teams of players compete in allocating limited resources to build up force structures consisting of notional weapon systems. The game consists of a number of turns, each turn representing one year. At the start of each "year," each side receives a budget, which it may allocate to acquisition and operation of various weapon systems and to other categories such as "intelligence" and "counterintelligence." Each year, conflict with the enemy becomes more probable, until a "war" occurs. At this point, each player's force structure is matched against the opponent's. Based on certain game rules, the more effective allocation wins.

The TEMPO Military Planning game is an important training tool, enabling future military decision makers to simulate making critical resource allocation decisions within an uncertain finite timeline. In courses given by the Defense Resource Management Institute for more than 40 years, more than 20,000 students from 125 countries have played the game as part of their training.

The research reported by Johnson et al. (2005) was aimed at applying computing power to resource allocation in a competitive environment. They undertook to develop a computer program capable of playing the TEMPO game (in a somewhat simplified version) against human opponents. By using a coevolutionary algorithm, they were able to create such a program based on a small set of fuzzy-logic decision rules. According to Johnson et al. (2005), coevolutionary algorithms consist of individuals against other individuals, each competing for resources in an environment that, in itself, poses its own threats. Competing individuals use random variation and selection to seek out survival strategies that will give them an edge over their opposition. (Johnson et al. 2005). These survival strategies were formulated on the principles of fuzzy logic, originally introduced by Zadeh (1965) through fuzzy set theory. According to Zadeh (1965), more often than not, the classes of objects found in the real physical world do not have precisely defined criteria of membership…clearly, "the class of beautiful women," or "the class of tall men"…are imprecisely defined classes. The imprecise classes provide for a framework that uses imprecise criteria to make decisions.

Johnson et al. (2005) claim near human-level performance for the program on the grounds that it was able to beat one of its developers on the first several tries. In what follows, TEMPO refers to this computerized version of the game.

1. Description (from "TEMPO Military Planning Game, Explanation and Rules for Players," p. 15–21)

a. Objective

Player wins a war by having more *Total Net Offensive (TNO) utils* (see below for description) than the opponent. War can occur in any given year with increasing probability of occurrence each year the game continues. *TNO utils* are only calculated if war occurs.

b. Overview

Weapon systems are grouped as follows:

- Type: Offensive (O) or Defensive (D)
- Class: A or B
- Level: 1, 2, 4, or 4

Weapon systems can take any combination of the above three levels of classification. For example, OA2, DB3, etc., are each valid weapon systems. Each weapon system produces an output of military capability that is valued in "UTILS." The UTIL is a measure of effectiveness used to simplify game playing and scoring. Although determining the effectiveness of weapons is often the most difficult part of military planning, this simplification permits the player to concentrate on the budget allocation problem. The player's goal is to allocate each weapon type so when war occurs they have more *TNO utils* than the opponent. Details on how this is accomplished follow.

There are four categories of actions the player can execute for each turn:

- ACQUIRE new weapon systems
- OPERATE existing inventory of weapon systems
- Purchase INTELligence
- Purchase COUNTER-INTELligence

An execution stage follows calculating and implementing these four decisions. The "execution stage" is defined as confirming all currently allocated decisions displayed on the screen and continuing to the next turn (i.e., next year). This execution stage is controlled by pressing the 'Commit' button.

(1) Operate. This function is how military performance is calculated (in the form of UTILS). Operating existing inventory results in a number of UTILS multiplied by the number of units within inventory operated. There are penalties for operating too much of any single system, in the form of diminishing returns, which

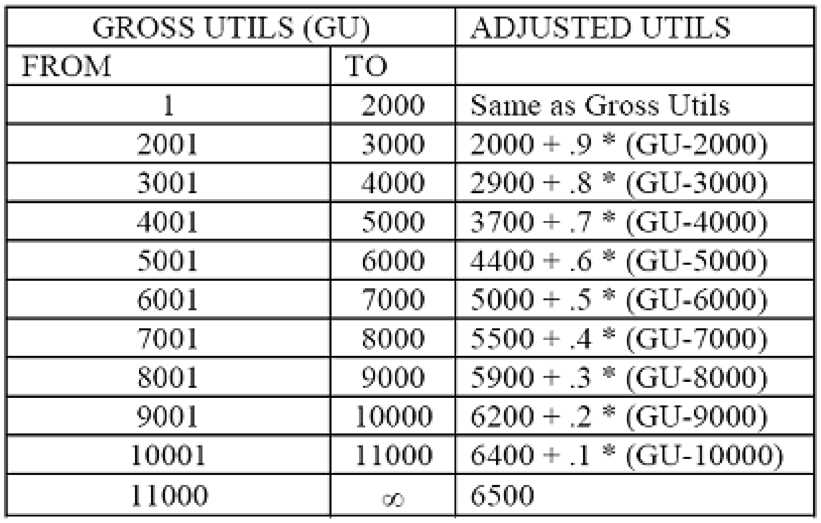
are discussed later. By operating a particular weapon, the inventory for that weapon remains for the next year. This is one method to ensure there are enough units for next year.

(2) Acquire. In order to OPERATE, units must exist in inventory. This is the other method to ensure there are enough units contained within inventory for the next year. Each year, a fixed amount of all available weapons are available to acquire.

(3) General. Only weapons existing in the inventory in the current year can be operated. Therefore, if the player decides not to operate any units in a given year, only those weapons the player acquires that year will be available within the inventory the next year. All other pre-existing weapon inventories are lost.

Only operating force units produce military capability and hence utils. New weapon systems do not necessarily replace existing systems; they might be additions to force structure. Util values for weapon systems remain constant throughout the game. Although util values per unit remain constant, if currently operated units in any one system produce more than 2000 utils, further utils are subject to "diminishing returns" on a sliding scale (see Figure 1). Costs remain constant during procurement and operation of a given system.

UTIL ADJUSTMENTS TO REFLECT DIMINISHING RETURNS
(FROM GROSS TO ADJUSTED UTILS)



GROSS UTILS (GU)		ADJUSTED UTILS
FROM	TO	
1	2000	Same as Gross Utils
2001	3000	2000 + .9 * (GU-2000)
3001	4000	2900 + .8 * (GU-3000)
4001	5000	3700 + .7 * (GU-4000)
5001	6000	4400 + .6 * (GU-5000)
6001	7000	5000 + .5 * (GU-6000)
7001	8000	5500 + .4 * (GU-7000)
8001	9000	5900 + .3 * (GU-8000)
9001	10000	6200 + .2 * (GU-9000)
10001	11000	6400 + .1 * (GU-10000)
11000	∞	6500

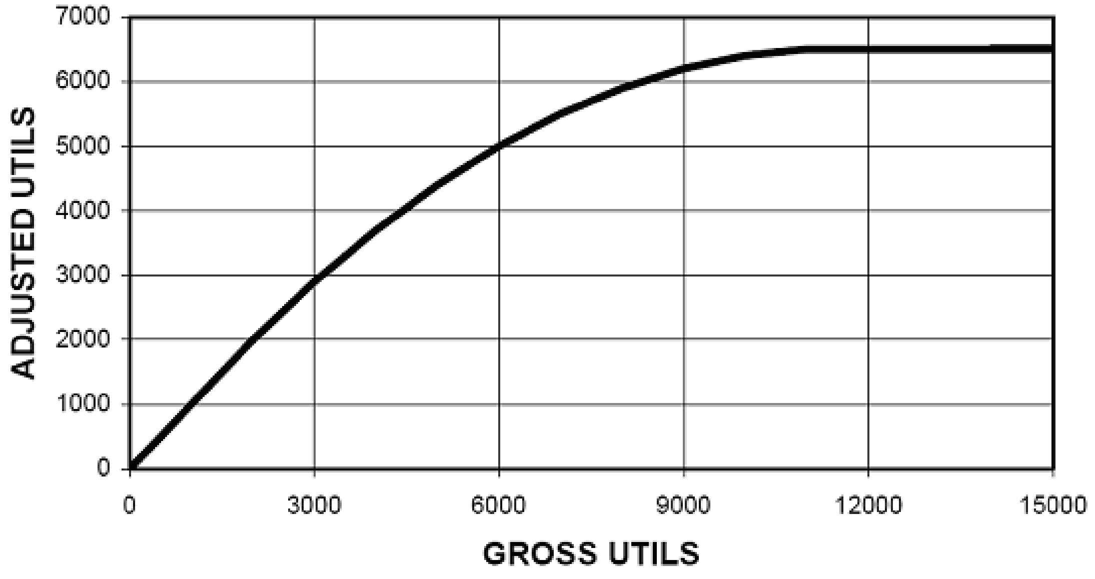


Figure 1. Implementation of Diminishing Returns (From: TEMPO Instructions)

Defensive weapon systems only defend against the opponent's offensive weapon systems of the same Force Unit Type; your DA weapons defend against only the enemy's OA weapons, and your DB weapons defend only against the enemy's OB weapons. These weapon systems cannot result in positive *TNO utils*.

Therefore, any DA system counts against any OA system of the enemy, and likewise for B systems, i.e., utils from operating DB2 defend against OB1, OB2, etc. However, once all OB1 utils are defended, the extra utils are useless. In other words, no credit is given for "overdefending" in a Force Unit Type category.

c. Scoring

A team's *TNO utils* for a particular year are calculated as follows:

$$\begin{aligned} HUM\_TNO_{UTILS\_j} &= \left(\sum_{i=1}^{n} HUM\_OA_i - \sum_{i=1}^{n} COMP\_DA_i \right)_j \\ &+ \left(\sum_{i=1}^{n} HUM\_OB_i - \sum_{i=1}^{n} COMP\_DB_i \right)_j \end{aligned} \tag{2.1}$$

$$\begin{aligned} COMP\_TNO_{UTILS\_j} &= \left(\sum_{i=1}^{n} COMP\_OA_i - \sum_{i=1}^{n} HUM\_DA_i \right)_j \\ &+ \left(\sum_{i=1}^{n} COMP\_OB_i - \sum_{i=1}^{n} HUM\_DB_i \right)_j \end{aligned} \tag{2.2}$$

where prefixes *HUM* and *COMP* denote 'human' and 'computer,' respectively. TNO_{UTILS_j} is the number of TNO Utils after year j. OA_i is the amount of utils derived from the type: offensive (*O*) weapon system class: *A*, level: *i*. OB_i, DA_i, and DB_i are similarly defined. In this game, *i* ranges from 1–4.

As mentioned previously, no credit is given for "overdefending" For example, if $\sum_{i=1}^{n} HUM\_DA_i$ is greater than $\sum_{i=1}^{n} COMP\_OA_i$ (both from first term in (2.2)) there is no additional benefit. In mathematical terms:

$$\sum_{i=1}^{n} HUM\_DA_i \;\; is\ replaced\ with \; \min\left\{ \sum_{i=1}^{n} HUM\_DA_i , \sum_{i=1}^{n} COMP\_OA_i \right\} \tag{2.3}$$

To win a war,

$$HUM\_TNO_{UTILS\_j} > COMP\_TNO_{UTILS\_j}\Big|_{j=WAR\_YEAR} \qquad (2.4)$$

d. Game Play

Normally, the game is played for several "years." War is inevitable and will eventually "break out." The amount of years before war occurs depends on a probability of war occurring. Each year this probability increases.

e. Probability and Results of War

The probability of war is announced at the beginning of each year and is for that year only. If war occurs, it only will happen after the budget decisions for that year have been completed but before the next year begins. Neither team can declare a war. If war occurs, the results of (2.1) and (2.2) will be computed, and the condition described in (2.4) will be evaluated. At this point the game ends and a winner is declared.

f. Budget

Each side has a finite budget. This budget increases each year by a random rate of increase. Mathematically, this is represented as:

$$BUDGET_{j+1} = (1+r)\cdot BUDGET_j \qquad (2.5)$$

where $r > 0$ is the random rate increase each year. Within the code, the randomized range of r is [0, 0.1]. $BUDGET_{j=0}$ = \$8,000. This budget expires at the end of each year similar to the DoD PPBS. There are four ways to allocate budget: (1) buy intelligence, (2) buy counter-intelligence, (3) acquire available weapon systems and (4) operate available inventory of weapon system.

g. Intelligence

Intelligence is divided into offensive and defensive intelligence. Offensive intelligence purchased in year j allows knowledge of the computer's offensive weapon systems adjusted utils of year *j* (displayed prior to year *j+1* commitment). Mathematically, these values are shown as: $\left(\sum_{i=1}^{n} COMP\_OA_i\right)_j$ and $\left(\sum_{i=1}^{n} COMP\_OB_i\right)_j$.

Similarly, defensive intelligence purchased allows knowledge of $\left(\sum_{i=1}^{n} COMP\_DA_i\right)_j$ and $\left(\sum_{i=1}^{n} COMP\_DB_i\right)_j$.

h. Counterintelligence

Counterintelligence ensures that if the computer buys intelligence, the information provided to them about the human player is less precise.

i. Acquisition

A team may acquire additional units of any system already in the inventory at any time. Units of any new system may be acquired when force information sheet indicates they are available and in any turn thereafter. In one or two (exceptional) cases you may get an initial inventory free of charge when the system first becomes available. The maximum annual acquisition rate for any particular system is stated.

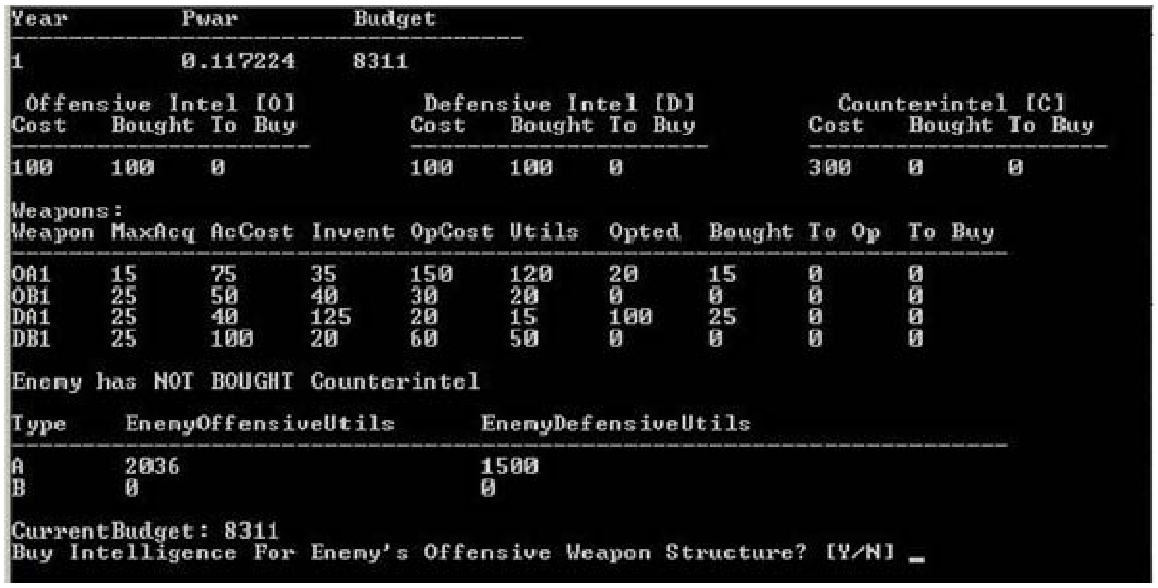
j. Operations

A team may operate any or all forces in inventory at the start of a year. Operation is not mandatory for any of the systems. One cannot “mothball” units. Existing force units not operated in any one year are lost from inventory. Units acquired in one year cannot be operated until the following year.

2. User Interface Evolution

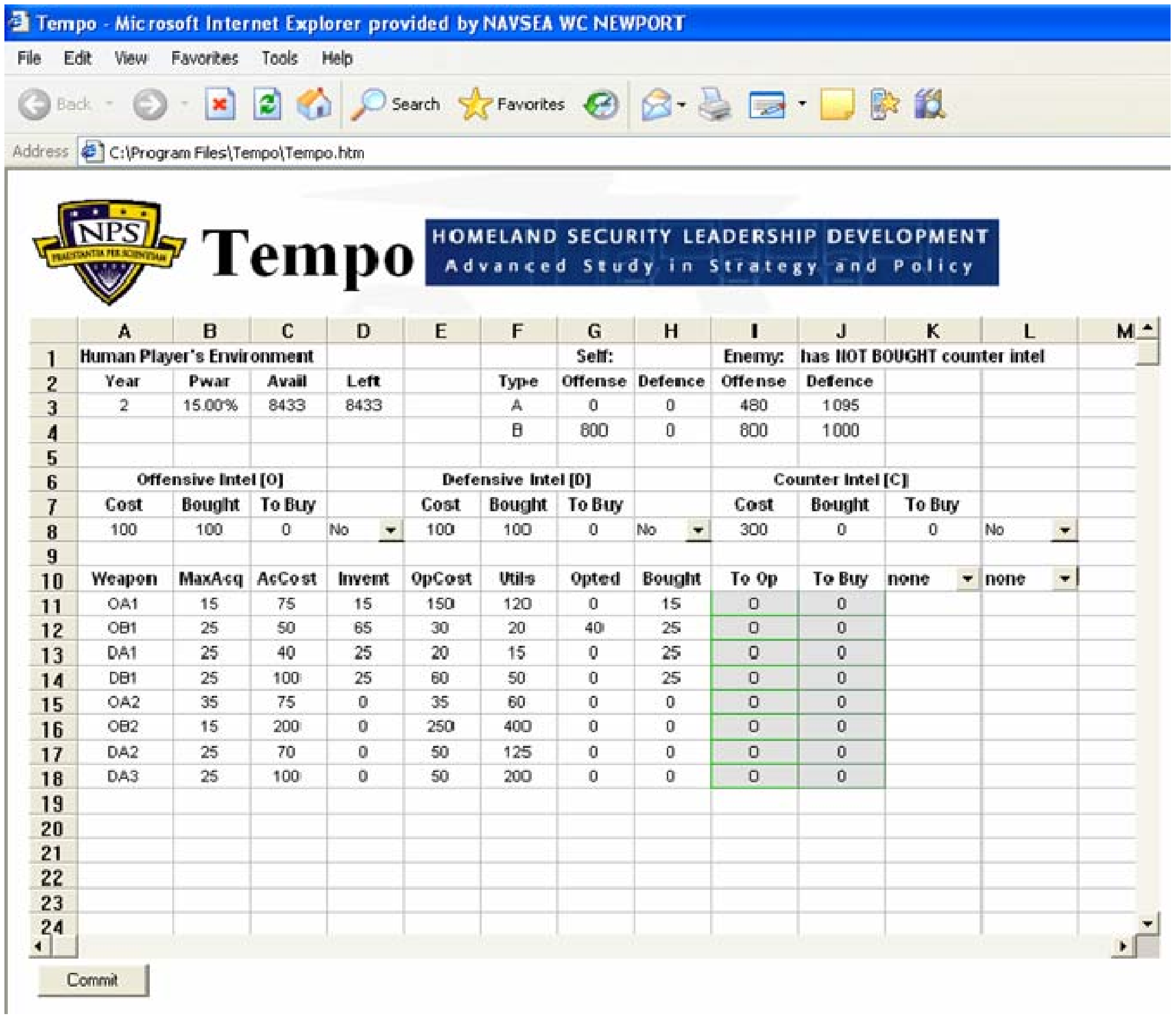
Figure 2 displays the original TEMPO user interface. The original TEMPO user interface was a command console where each allocation was required to be submitted one at a time. This method did not accommodate a change of mind—once an allocation was entered on a weapon type, it was unchangeable. This makes it difficult to properly plan the budget.

Figure 3 through Figure 7 display and describe the modified TEMPO user interface. This version relied on Active Template Library (ATL) code. The code associated with this version developed a grid Microsoft Component Object Model (COM) Object as the interface, which was loaded within Microsoft's Internet Explorer™. This modified TEMPO version allows the user to make decisions, determine impacts on cost and utils, and change things prior to committing the budget. This is a major enhancement to implementing a realistic application of the resource allocation process; however, this TEMPO version still retains many limitations along with the original TEMPO command console interface for successful SEVER implementation. These limitations are discussed further in the next chapter.



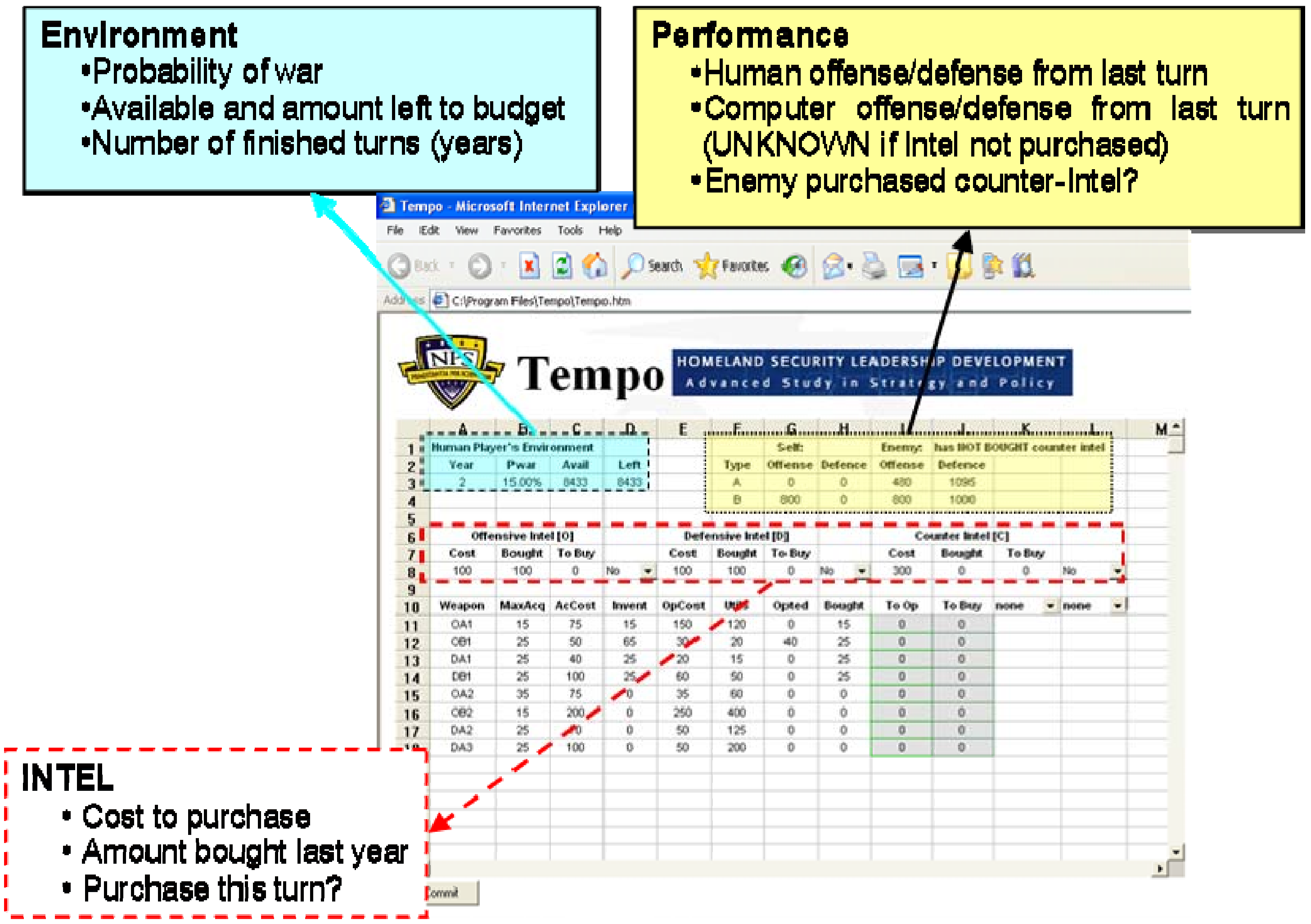
```
Year         Pwar         Budget
-----------------------------------------
1            0.117224     8311

 Offensive Intel [O]         Defensive Intel [D]         Counterintel [C]
Cost    Bought To Buy        Cost    Bought To Buy        Cost    Bought To Buy
----------------------        ----------------------       ----------------------
100     100    0              100     100    0             300     0      0

Weapons:
Weapon MaxAcq  AcCost  Invent  OpCost  Utils   Opted   Bought  To Op   To Buy
------------------------------------------------------------------------------
OA1    15      75      35      150     120     20      15      0       0
OB1    25      50      40      30      20      0       0       0       0
DA1    25      40      125     20      15      100     25      0       0
DB1    25      100     20      60      50      0       0       0       0

Enemy has NOT BOUGHT Counterintel

Type     EnemyOffensiveUtils       EnemyDefensiveUtils
------------------------------------------------------------------------------
A        2036                      1500
B        0                         0

CurrentBudget: 8311
Buy Intelligence For Enemy's Offensive Weapon Structure? [Y/N] _
```

Figure 2. Original TEMPO User Interface


Tempo - Microsoft Internet Explorer provided by NAVSEA WC NEWPORT
File Edit View Favorites Tools Help
Back Search Favorites
Address C:\Program Files\Tempo\Tempo.htm
NPS
Tempo
HOMELAND SECURITY LEADERSHIP DEVELOPMENT
Advanced Study in Strategy and Policy
Commit

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Human Player's Environment						Self:		Enemy:	has NOT BOUGHT counter intel			
2	Year	Pwar	Avail	Left		Type	Offense	Defence	Offense	Defence			
3	2	15.00%	8433	8433		A	0	0	480	1095			
4						B	800	0	800	1000			
5													
6		Offensive Intel [O]				Defensive Intel [D]				Counter Intel [C]			
7	Cost	Bought	To Buy		Cost	Bought	To Buy		Cost	Bought	To Buy		
8	100	100	0	No	100	100	0	No	300	0	0	No	
9													
10	Weapon	MaxAcq	AcCost	Invent	OpCost	Utils	Opted	Bought	To Op	To Buy	none	none	
11	OA1	15	75	15	150	120	0	15	0	0			
12	OB1	25	50	65	30	20	40	25	0	0			
13	DA1	25	40	25	20	15	0	25	0	0			
14	DB1	25	100	25	60	50	0	25	0	0			
15	OA2	35	75	0	35	60	0	0	0	0			
16	OB2	15	200	0	250	400	0	0	0	0			
17	DA2	25	70	0	50	125	0	0	0	0			
18	DA3	25	100	0	50	200	0	0	0	0			
19													
20													
21													
22													
23													
24													

Figure 3. Updated TEMPO Graphical User Interface



Environment
- •Probability of war
- •Available and amount left to budget
- •Number of finished turns (years)

Performance
- •Human offense/defense from last turn
- •Computer offense/defense from last turn (UNKNOWN if Intel not purchased)
- •Enemy purchased counter-Intel?

INTEL
- Cost to purchase
- Amount bought last year
- Purchase this turn?

Tempo - Microsoft Internet Explorer

C:\Program Files\Tempo\Tempo.htm

Tempo

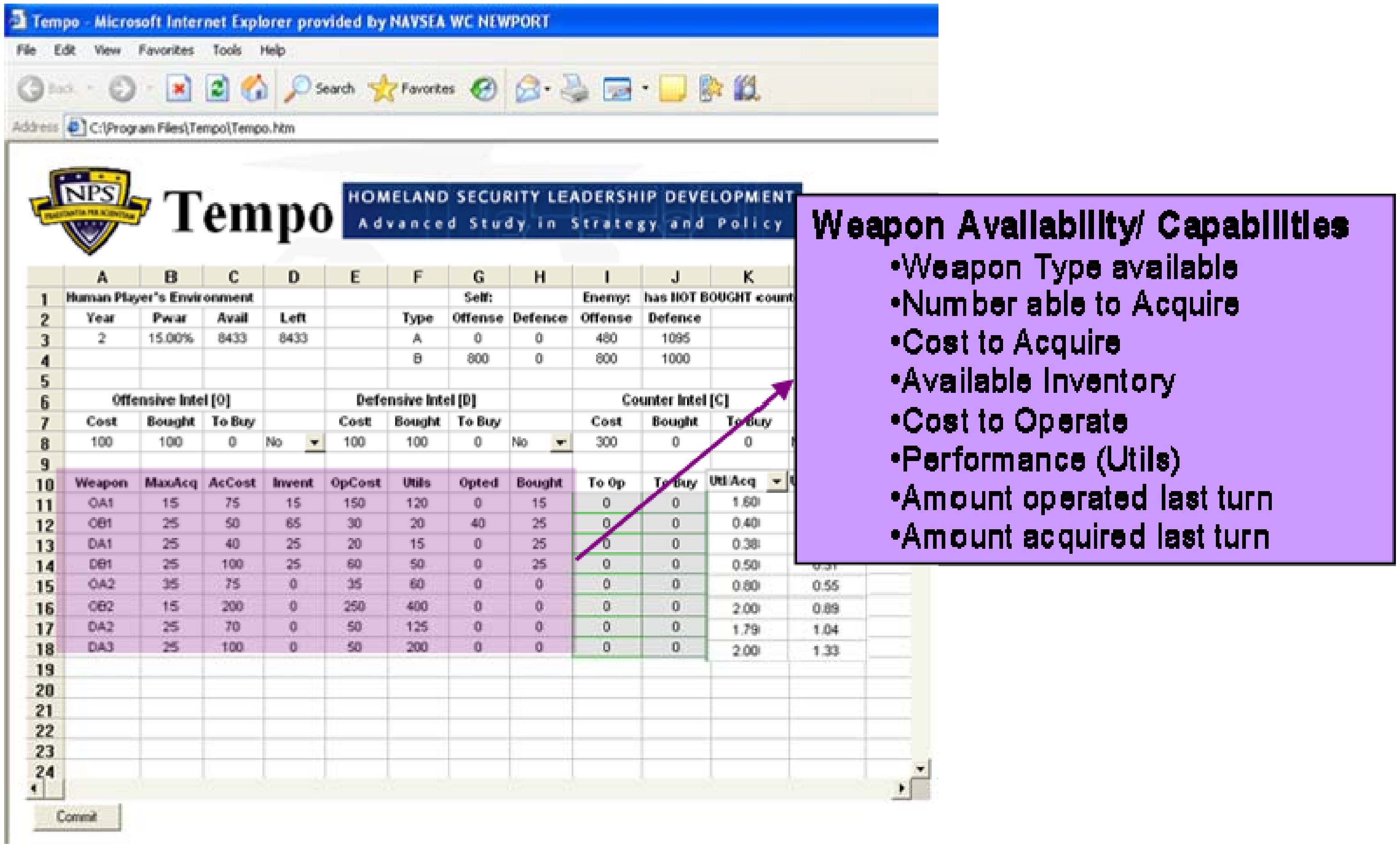
HOMELAND SECURITY LEADERSHIP DEVELOPMENT

Advanced Study in Strategy and Policy

	A	B	C	D	E	F	G	H	I	J	K	L
1	Human Player's Environment						Self:		Enemy:	has NOT BOUGHT counter intel		
2	Year	Pwar	Avail	Left		Type	Offense	Defence	Offense	Defence		
3	2	15.00%	8433	8433		A	0	0	480	1095		
4						B	800	0	800	1000		
5												
6		Offensive Intel [O]				Defensive Intel [D]				Counter Intel [C]		
7	Cost	Bought	To Buy		Cost	Bought	To Buy		Cost	Bought	To Buy	
8	100	100	0	No	100	100	0	No	300	0	0	No
9												
10	Weapon	MaxAcq	AcCost	Invent	OpCost	Util	Opted	Bought	To Op	To Buy	none	none
11	OA1	15	75	15	150	120	0	15	0	0		
12	OB1	25	50	65	30	20	40	25	0	0		
13	DA1	25	40	25	20	15	0	25	0	0		
14	DB1	25	100	25	60	50	0	25	0	0		
15	OA2	35	75	0	35	60	0	0	0	0		
16	OB2	15	200	0	250	400	0	0	0	0		
17	DA2	25	70	0	50	125	0	0	0	0		
18	DA3	25	100	0	50	200	0	0	0	0		

Commit

Figure 4. Breakout of Existing *Environment Information* Display on the Updated TEMPO GUI

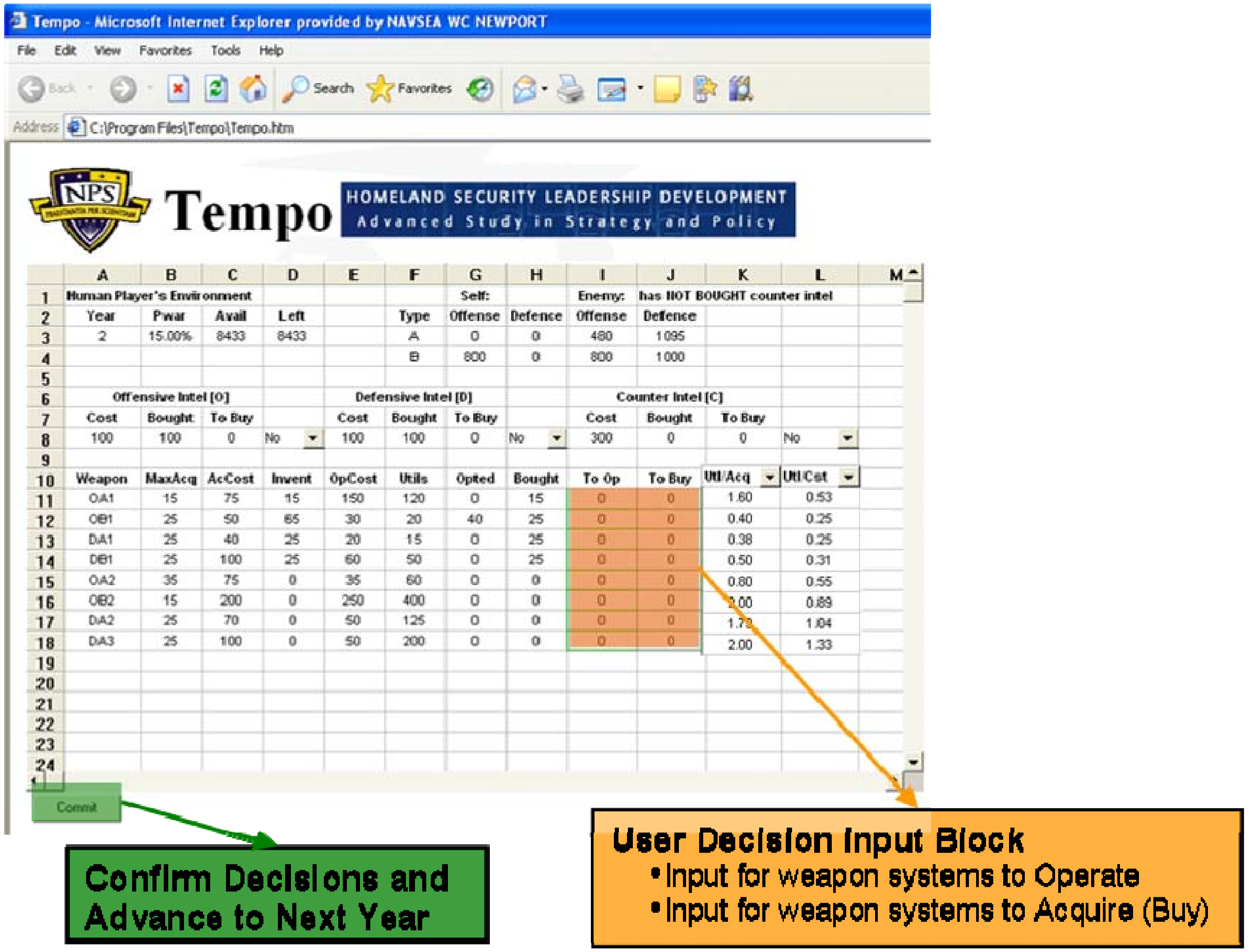


Tempo - Microsoft Internet Explorer provided by NAVSEA WC NEWPORT
File Edit View Favorites Tools Help
Back Search Favorites
Address C:\Program Files\Tempo\Tempo.htm
Tempo
HOMELAND SECURITY LEADERSHIP DEVELOPMENT
Advanced Study in Strategy and Policy
Commit

	A	B	C	D	E	F	G	H	I	J	K	L
1	Human Player's Environment						Self:		Enemy:	has NOT BOUGHT count		
2	Year	Pwar	Avail	Left		Type	Offense	Defence	Offense	Defence		
3	2	15.00%	8433	8433		A	0	0	480	1095		
4						B	800	0	800	1000		
5												
6	Offensive Intel [O]				Defensive Intel [D]				Counter Intel [C]			
7	Cost	Bought	To Buy		Cost	Bought	To Buy		Cost	Bought	To Buy	
8	100	100	0	No	100	100	0	No	300	0	0	
9												
10	Weapon	MaxAcq	AcCost	Invent	OpCost	Utils	Opted	Bought	To Op	To Buy	Utl/Acq	
11	OA1	15	75	15	150	120	0	15	0	0	1.60	
12	OB1	25	50	65	30	20	40	25	0	0	0.40	
13	DA1	25	40	25	20	15	0	25	0	0	0.38	
14	DB1	25	100	25	60	50	0	25	0	0	0.50	[illegible]
15	OA2	35	75	0	35	60	0	0	0	0	0.80	0.55
16	OB2	15	200	0	250	400	0	0	0	0	2.00	0.89
17	DA2	25	70	0	50	125	0	0	0	0	1.79	1.04
18	DA3	25	100	0	50	200	0	0	0	0	2.00	1.33
19												
20												
21												
22												
23												
24												

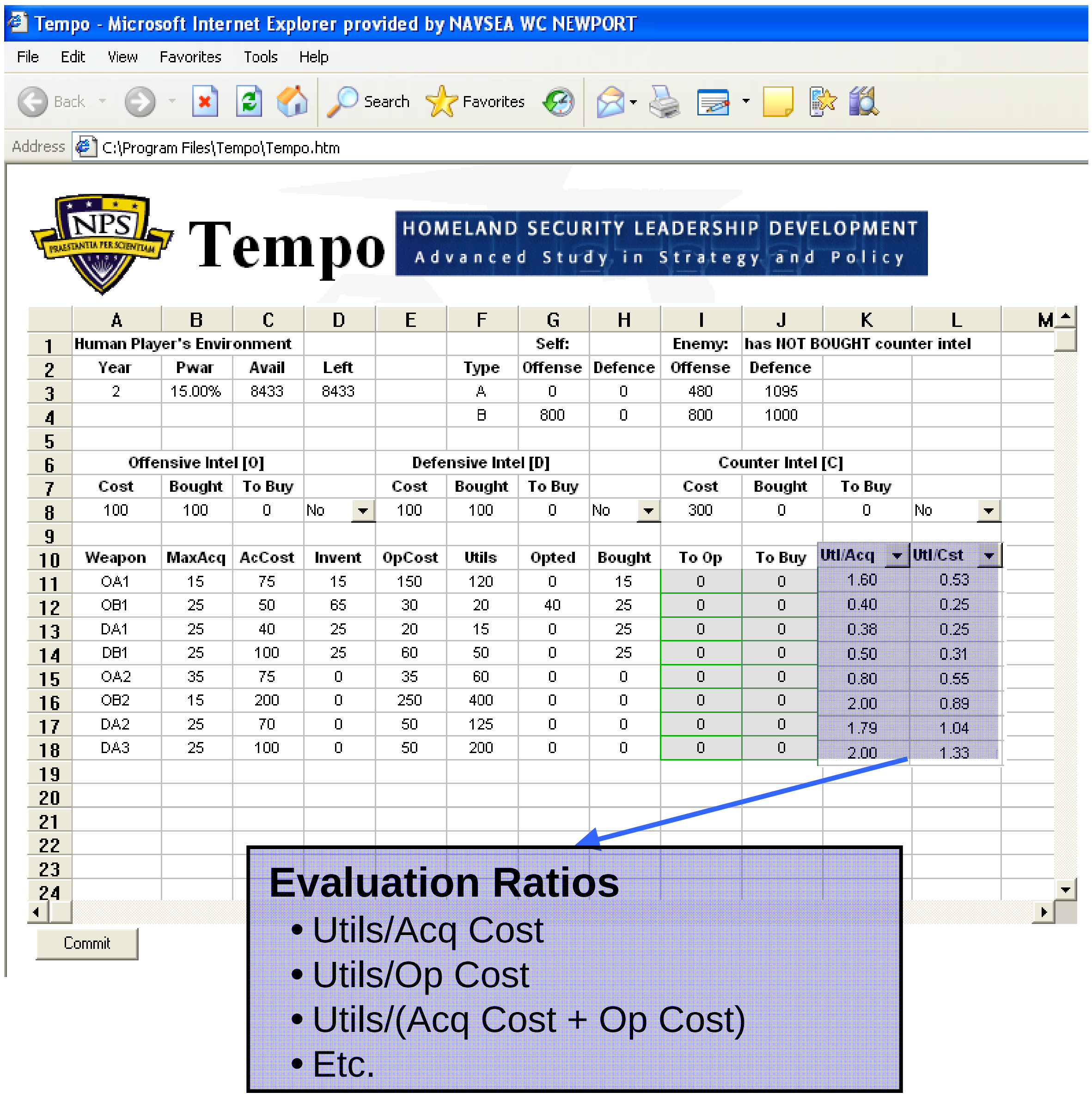
Weapon Availability/ Capabilities
- Weapon Type available
- Number able to Acquire
- Cost to Acquire
- Available Inventory
- Cost to Operate
- Performance (Utils)
- Amount operated last turn
- Amount acquired last turn

Figure 5. Breakout of Existing *Weapon System Information* on the Updated TEMPO GUI


Tempo - Microsoft Internet Explorer provided by NAVSEA WC NEWPORT
File Edit View Favorites Tools Help
Back Search Favorites
Address C:\Program Files\Tempo\Tempo.htm
Tempo
HOMELAND SECURITY LEADERSHIP DEVELOPMENT
Advanced Study in Strategy and Policy
Commit
Confirm Decisions and Advance to Next Year
User Decision Input Block
• Input for weapon systems to Operate
• Input for weapon systems to Acquire (Buy)

	A	B	C	D	E	F	G	H	I	J	K	L
1	Human Player's Environment						Self:		Enemy:	has NOT BOUGHT counter intel		
2	Year	Pwar	Avail	Left		Type	Offense	Defence	Offense	Defence		
3	2	15.00%	8433	8433		A	0	0	480	1095		
4						B	800	0	800	1000		
5												
6		Offensive Intel [O]				Defensive Intel [D]				Counter Intel [C]		
7	Cost	Bought	To Buy		Cost	Bought	To Buy		Cost	Bought	To Buy	
8	100	100	0	No	100	100	0	No	300	0	0	No
9												
10	Weapon	MaxAcq	AcCost	Invent	OpCost	Utils	Opted	Bought	To Op	To Buy	Utl/Acq	Utl/Cst
11	OA1	15	75	15	150	120	0	15	0	0	1.60	0.53
12	OB1	25	50	65	30	20	40	25	0	0	0.40	0.25
13	DA1	25	40	25	20	15	0	25	0	0	0.38	0.25
14	DB1	25	100	25	60	50	0	25	0	0	0.50	0.31
15	OA2	35	75	0	35	60	0	0	0	0	0.80	0.55
16	OB2	15	200	0	250	400	0	0	0	0	2.00	0.89
17	DA2	25	70	0	50	125	0	0	0	0	1.7[illegible]	1.04
18	DA3	25	100	0	50	200	0	0	0	0	2.00	1.33
19												
20												
21												
22												
23												
24												

Figure 6. Breakout of *Decision Execution* Interfaces on the updated TEMPO GUI


Tempo - Microsoft Internet Explorer provided by NAVSEA WC NEWPORT
File Edit View Favorites Tools Help
Back Search Favorites
Address C:\Program Files\Tempo\Tempo.htm
NPS PRAESTANTIA PER SCIENTIAM
Tempo
HOMELAND SECURITY LEADERSHIP DEVELOPMENT
Advanced Study in Strategy and Policy
Commit

	A	B	C	D	E	F	G	H	I	J	K	L
1	Human Player's Environment						Self:		Enemy:	has NOT BOUGHT counter intel		
2	Year	Pwar	Avail	Left		Type	Offense	Defence	Offense	Defence		
3	2	15.00%	8433	8433		A	0	0	480	1095		
4						B	800	0	800	1000		
5												
6	Offensive Intel [O]				Defensive Intel [D]				Counter Intel [C]			
7	Cost	Bought	To Buy		Cost	Bought	To Buy		Cost	Bought	To Buy	
8	100	100	0	No	100	100	0	No	300	0	0	No
9												
10	Weapon	MaxAcq	AcCost	Invent	OpCost	Utils	Opted	Bought	To Op	To Buy	Utl/Acq	Utl/Cst
11	OA1	15	75	15	150	120	0	15	0	0	1.60	0.53
12	OB1	25	50	65	30	20	40	25	0	0	0.40	0.25
13	DA1	25	40	25	20	15	0	25	0	0	0.38	0.25
14	DB1	25	100	25	60	50	0	25	0	0	0.50	0.31
15	OA2	35	75	0	35	60	0	0	0	0	0.80	0.55
16	OB2	15	200	0	250	400	0	0	0	0	2.00	0.89
17	DA2	25	70	0	50	125	0	0	0	0	1.79	1.04
18	DA3	25	100	0	50	200	0	0	0	0	2.00	1.33
19												
20												
21												
22												
23												
24												

Evaluation Ratios
- Utils/Acq Cost
- Utils/Op Cost
- Utils/(Acq Cost + Op Cost)
- Etc.

Figure 7. Breakout of *Decision Evaluation* Information on the Updated TEMPO GUI

B. SYSTEMS ENGINEERING VALUE EQUATION WITH RISK (SEVER)

SEVER was derived and first implemented by Langford (2006) under the process of 'Rapid Systems Engineering,' Now termed Value Systems Engineering. Value Systems Engineering is a scenario-driven approach that attempts to reduce the degree of uncertainty in predicting business success by structuring and analyzing the interplay between alternative business models, competitive strategies, and their resultant product alternatives. It is a bottom-up, systematic, and highly iterative approach relating competitive strategies to alternative business requirements and conditions for stakeholder success (Langford, 2006). Langford (2006) utilizes SEVER as the fundamental tool to quantitatively predict success of particular businesses. SEVER is closely related to Value Analysis and Engineering (Miles 1972). Both view a system as a collection of functions arranged in a hierarchical fashion. Details on Value Systems Engineering are located in Appendix A.

THIS PAGE INTENTIONALLY LEFT BLANK

III. DEVELOPMENT METHODS

In order to address the fundamental research question in this thesis, both a thorough understanding and detailed decomposition of the TEMPO model and the SEVER algorithm were required. Specifically, once this decomposition process began, the challenge undertaken was how to properly represent TEMPO output variables as SEVER input variables. Once achieved, more detailed questions surfaced. Answers to some of these questions were provided through actual software development. These questions include:

What is required?

- Represent performance, investment, function, and quality, as defined by SEVER, from TEMPO variables.
- Determine weapon availability patterns.
- Understand pWar and the value it provides. Is this realistic?
- Characterize the computer's weapon allocation decisions. Are these decisions consistent for each year from one game to the next?
- Identify TEMPO software modifications required to incorporate the SEVER outputs.

How can this be achieved?

- Conduct many trials of the game.
- Capture each trial's output permanently.
- Develop particular strategies to play against the computer.
- Post-process trial data to determine the effectiveness of certain strategies (used to compare against the predicted SEVER value for each strategy, once implemented).

Where possible, the answers to these questions are provided in this chapter. Other questions that could not be answered provide the framework for future work.

This thesis describes the SEVER algorithm, yet no formal execution of the algorithm or related software execution is presented. The work presented is only for a software package that will be used in the future to evaluate SEVER.

A. APPROACH

SEVER in its entirety is reformulated for each function and represented as:

$$\frac{Risk}{Threat} = \frac{Vulnerability \bullet Performance \bullet Quality}{Investment} \tag{3.1}$$

All terms are consistent with the definitions previously given. Regarding the units of each variable, *performance* is expressed in the same units as 1/*threat* (i.e., for TEMPO, ‘Utils’). Quality is expressed as the same units of investment (i.e., for TEMPO, ‘$’). *Vulnerability* is a probability without units. This defines the expression $\frac{Risk}{Threat}$ in units of *performance* (i.e., for TEMPO: ‘Utils’). In plain English, this representation states the ratio of the risk to all threats that would prevent the performing a particular system function is equivalent to (all else remaining constant):

1. *How vulnerable is the function to the composite threat*—if the function is invulnerable to the threat, the risk of any threat to eliminate it is negligible

2. *How well the function performs*—if the function does not perform well, the risk of any threat to eliminate it is negligible

3. *How much of a value is the function to society*—if the function’s quality (i.e., value to society) is high, the risk of any threat to eliminate it is high

4. *How inexpensive was the function to develop*—if the function was cheap to develop and implement, the risk of any threat to eliminate it is low since it can:

    a. be developed or implemented again

    b. provided redundantly

In order to calculate the total system *value*, each function’s *value* must be quantitatively derived. Thus, applying SEVER to a system involves a top-down system decomposition (collected using the functional decomposition method) and a bottom-up calculation of each function’s *value*. The final product is a summation of all the system’s function *values*.

According to Langford (2006), combined with the investment made by a stakeholder, functions, performance, and quality requirements determine the operational capability of the product or service. Thus, considering a product or service as a system, the framework for calculating a system’s value is represented as the system’s operational capability given a certain required investment. Multiple systems attempting to achieve the same end goal can be compared using this framework. The system’s operational capability, or *worth*, is based on its functional view and is quantitatively measured in the same units as *performance*.

Further, this framework can be extended to evaluate strategic decisions, or more appropriate for this thesis, a strategy. If a strategy is thought of as a system (i.e., the output of a particular strategy provides more value to society than others—society in this case is the player), the value of one strategy against another can be compared analogously to products and services using SEVER.

1. Decomposed for TEMPO Application

The following decomposition explains the how the calculation of *consequence* within the context of TEMPO is captured. By more formally representing *investment* algebraically as:

$$I = \left[\frac{c}{t_{unit}}\right] \bullet T_{Total} \tag{3.2}$$

where c is the incremental unit cost, t_{unit} is the smallest applicable increment measure of time, and T_{Total} is a total aggregate measure of time for a function, *value per function* can now be expressed as:

$$\frac{V}{F}=\left[\frac{P_{incremental}}{\frac{c}{t_{unit}}\bullet T_{Total}}\right]\bullet Q_{Total} \tag{3.3}$$

In TEMPO, $\sum_{i=1}^{n} t_{unit_i} = T_{Total}$. t_{unit} represents one year and T_{Total} represents the total duration of the game. Units for *I* are '\$.' Rearranging terms and multiplying by P_{Total} / P_{Total} yields:

$$\frac{V}{F}=\left[\frac{P_{incremental}}{\frac{c}{t_{unit}}}\right]\bullet\left[\frac{P_{Total}}{T_{Total}}\right]\bullet\left[\frac{Q_{Total}}{P_{Total}}\right] \tag{3.4}$$

$P_{incremental}$ represents the performance for a function per year, where P_{Total} represents the total performance over the course of the game. Similarly, Q_{Total} is measured as the total quality of the function over the course of the game. This is the representation currently considered for use with TEMPO. It consists of a product of three major terms to align with the particular outputs expected from TEMPO. V/F is calculated in the same units of P, *Utils*.

Q is a quality function measured as a loss to society (Taguchi, 1990) represented by (o.10) below. The units of Q are the same as I: '\$'.

$$Q(P)=\beta-L_n(P) \tag{3.5}$$

where $L_n(P)$ represents the loss function. Typically, quality as a function of performance is estimated using a quadratic loss function. A deviation from the optimal performance results in a loss equivalent to that deviation squared. See Figure 8 for the relationship between quality and performance. Qualitatively, β is the maximum theoretical measure of quality corresponding to the optimal performance level. This value represents both the developer and the user.

a. Derivation of Taguchi Loss Function

Consider a representation of quality as shown in (3.6):

$$Q_m + L_T\left(Y\right) = B \tag{3.6}$$

where B is the maximum quality, Q_m is the measured quality, and $L_T(Y)$ is the Taguchi Loss function, as stated in (3.7),

$$L_T\left(Y\right) = k\left(Y - m\right)^2 \tag{3.7}$$

where k is the loss factor (derived empirically), Y is the individually measured performance, and m is the ideal target performance. k is further decomposed as:

$$k = \frac{A}{\delta^2} \tag{3.8}$$

where A is the average cost of each unit returned/reworked and δ is the range of variability in performance:

$$\delta = \left(UL - LL\right)/2 \tag{3.9}$$

UL and LL are the upper and lower limits away from the ideal target performance, m, expected.

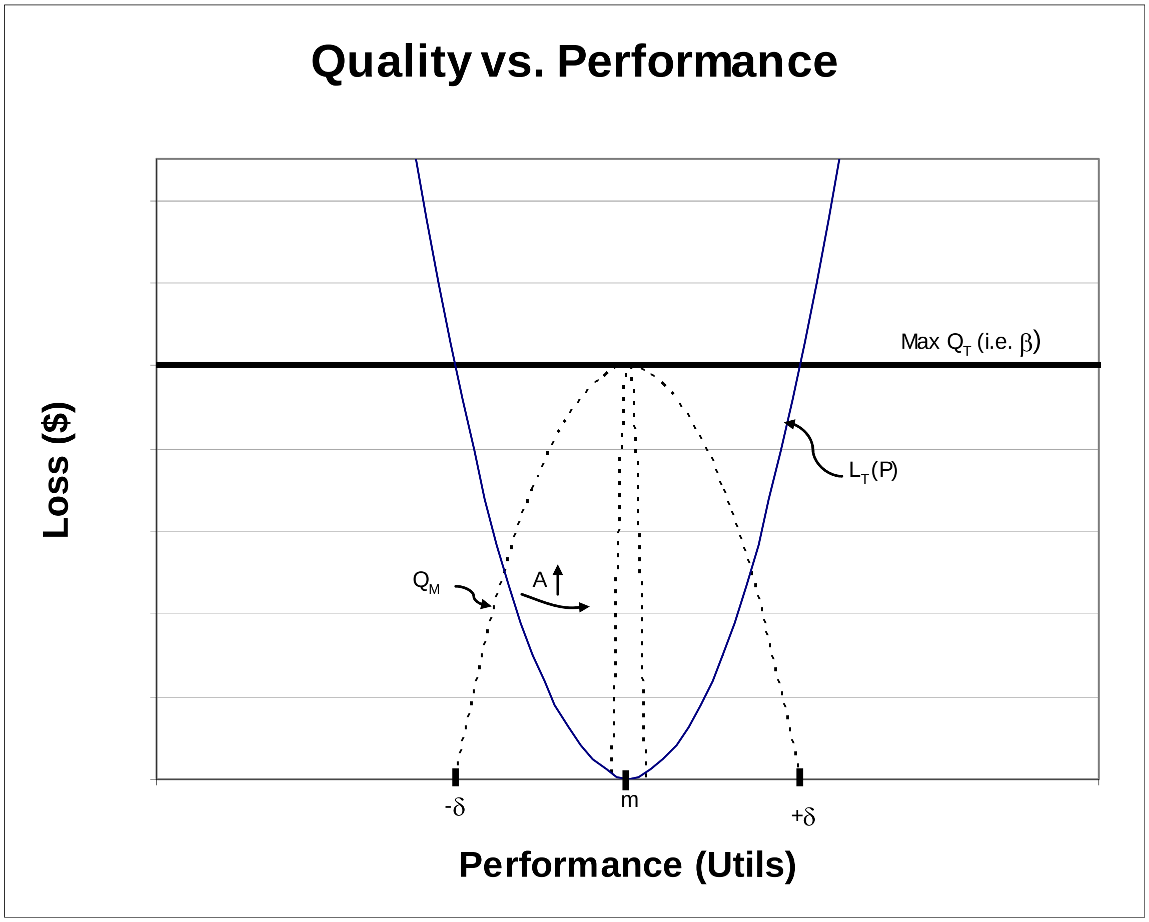


Figure 8. Generic Representation of *Quality* as a Function of *Performance*

Thus, quality is represented completely as shown in (3.10):

$$Q_T = \beta - \frac{A}{\left(\frac{UL - LL}{2}\right)^2}(Y - m)^2 \tag{3.10}$$

b. Commentary

From this point, properly deriving the quality of each decision in the TEMPO game requires empirical effort based on statistical data. TEMPO Version 3 is now capable of performing the groundwork analysis to derive the quality of each decision. This empirical analysis to further define quality of each decision remains a future effort.

TEMPO Version 3 involved the following development:

- Creating an automatic gameplay feature that allows a defined number of games to be played with particular strategies.
- Developing the rule-based resource allocation strategies
- Updating the GUI to include SEVER outputs
- Conducting many games to prove the output of pWar is accurate

These new capabilities enabled TEMPO to automatically conduct a user-defined number of trials for SEVER to be statistically evaluated. The variables required for SEVER can now be empirically derived and calibrated to prove that the value determined by SEVER is accurate for a particular resource allocation strategy employed. The TEMPO Version 3 software development effort is described in greater detail in the next chapter.

B. EVALUATION

Based on this research, the method incorporated to perform preliminary testing included coding rule-based strategies to capture a more detailed understanding of TEMPO and its potential application to SEVER. The ultimate goal is to allow SEVER to score each resource allocation decision 'on-the-fly' prior to the player committing his/her decisions. This can only be accomplished once the value and risks for a given decision are understood and quantified.

Six different rule-based strategy types were developed. Of those, a total of 10 individual strategies were executed. Turn-by-turn data was collected for each execution for a total of 1000 trials. Each strategy was stored as a separate MATLAB workspace file. A MATLAB function, "collectStats.m" collected, organized, and stored these separate workspace files. The MATLAB function, "compOutput.m" computed the output graphs for each strategy used to compare the results for each strategy. The philosophy in developing this code, along with the previously described MATLAB software, is to design in the flexibility to implement, run, and evaluate new strategies as they are derived.

A box plot is used to present a high-level comparison between each strategy. The MATLAB function, ‘boxplot.m’ (part of the Statistics Toolbox) was used within the ‘compOutput.m’ function to generate Figure 29, page 74. This plot displays each strategy’s “player net utils when war occurs” over 1000 trails. Values above 0 are player wins. Values below 0 are computer wins. From this plot, a statistical comparison of each strategy’s performance is presented. The data is represented in a box and whisker format. The box represents the 25th to 75th percentiles; the line within the box represents the median; the whiskers represent the rest of the data not considered outliers; and the individual plotted points (+’s) represent outliers, as determined by the MATLAB function.

IV. DEVELOPMENT, RESULTS, AND RECOMMENDATIONS

This chapter describes the development of the software. A functional description of the new software package is provided. This description is separated into six major sections: an overview of the software package, the software integration challenges, TEMPO Version 3 development from TEMPO Version 2, MATLAB strategy development, the MATLAB analysis development, and the total integration. New features were considered and implemented based upon the inputs required by the SEVER algorithm, the existing capabilities of TEMPO, and the measurable outputs required by MATLAB to run the rule-based strategies and analyze all strategies.

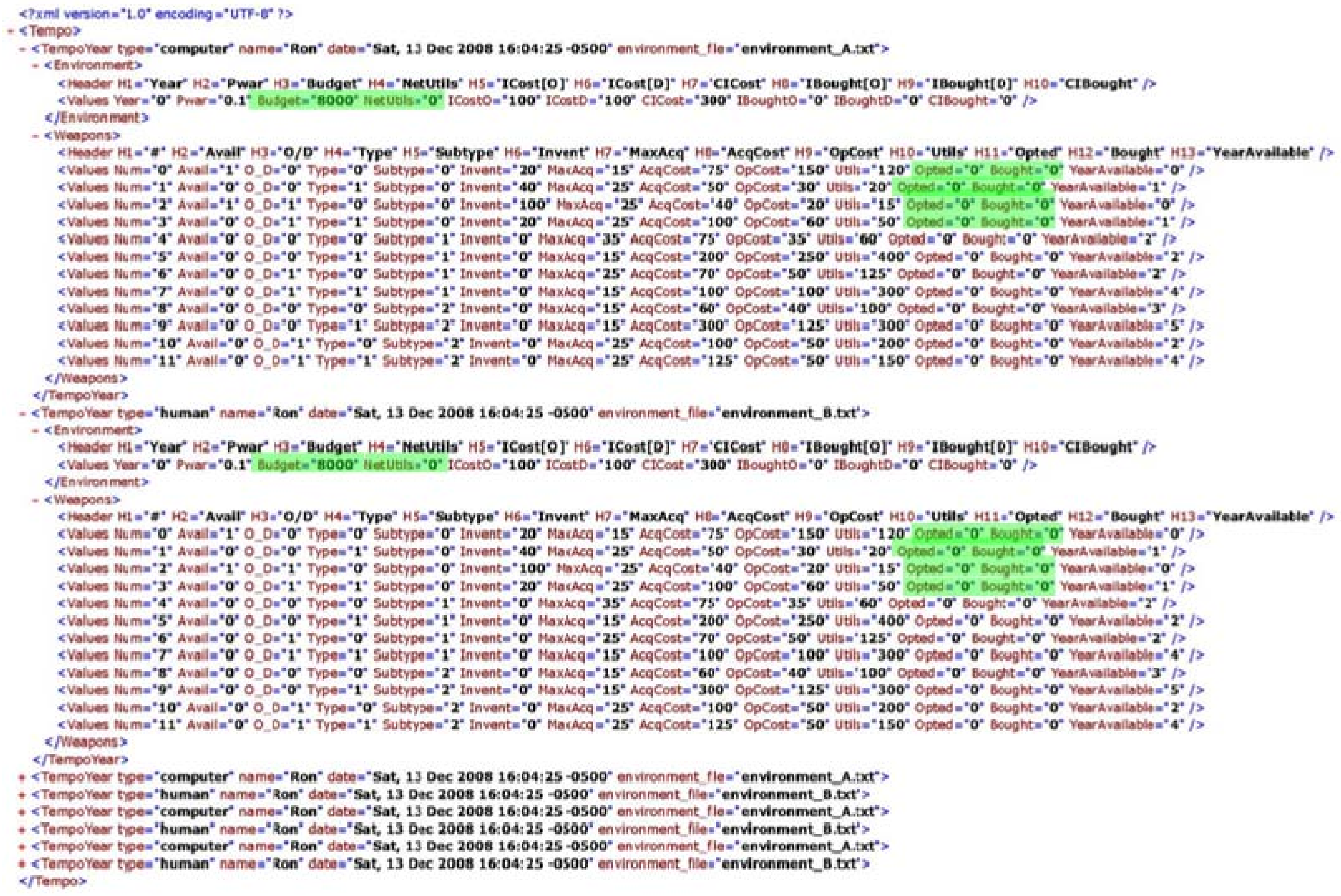
A. OVERVIEW OF THE SOFTWARE PACKAGE

The new software consists of code written in both TEMPO.NET and MATLAB. This software package was designed to execute particular strategies within TEMPO.NET and capture/present the results for a defined number of trials. Executing a strategy consists of the user selecting a desired strategy and number of trials to execute. Capturing the resulting output data required a TEMPO.ATL model update and MATLAB software development.

There were two major features required for TEMPO to gather the appropriate measures to implement SEVER. The first feature was for TEMPO to automatically write an output log file at the end of each game so this data could be permanently stored. The second feature of TEMPO.NET is the ability to automatically play a defined number of games.

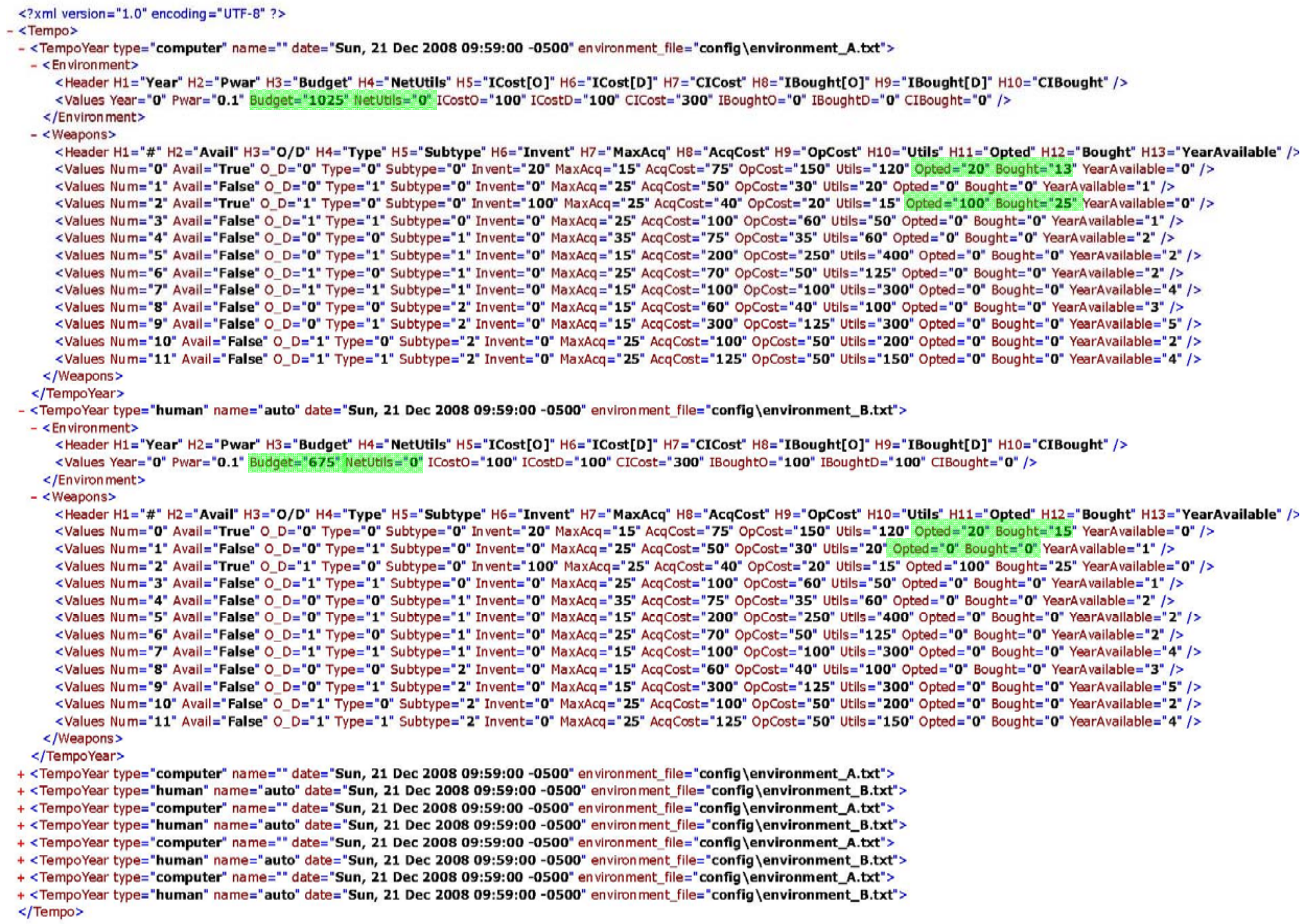
The first feature already existed in the existing TEMPO ATL version. A log file for each game was provided in an eXtensible Markup Language (XML) format. Examples of this log file for a typical game’s first year is presented in Figure 9 and for a typical game’s last year is presented in Figure 11. The highlighted fields are those that required modification. In order to store the appropriate data for analysis, some modifications to the data output in this file were required. All weapon inventories and maximum numbers to purchase were incorrectly listed. For example, Year *n+1* data was listed where Year *n* data should be. This is believed to be a bug in the existing ATL version and was updated by: (1) storing the previous year's weapon data, (2) executing that year's turn, and (3) writing the stored 'inventory/maximum acquisition' (or, weapon availability) data to the log file (along with the current year's allocations) after committing to those weapon allocation decisions. Without incorporating this fix, comparison between weapon availability data and the allocation decision for every year would not be possible. The data associated with the first year's weapon availability is lost and the data associated with the year that war occurred is unnecessary as it reflects weapon availabilities after war already occurred.

The second update was implemented to facilitate data analysis, but it was not mandatory. It consisted of displaying the budget left after the turn was executed, rather than the original budget for the start of the turn. This was convenient for the MATLAB analysis tool, as the actual budget left is considered important rather than the starting budget. In either case, the starting budget could be calculated assuming the weapons operated and acquired for that turn were accurate (as accomplished by the first update) or vice versa. Associated samples of the new log file are shown in Figure 10 and Figure 12, for a typical game's first and last year, respectively. The modified fields are highlighted for clarity.



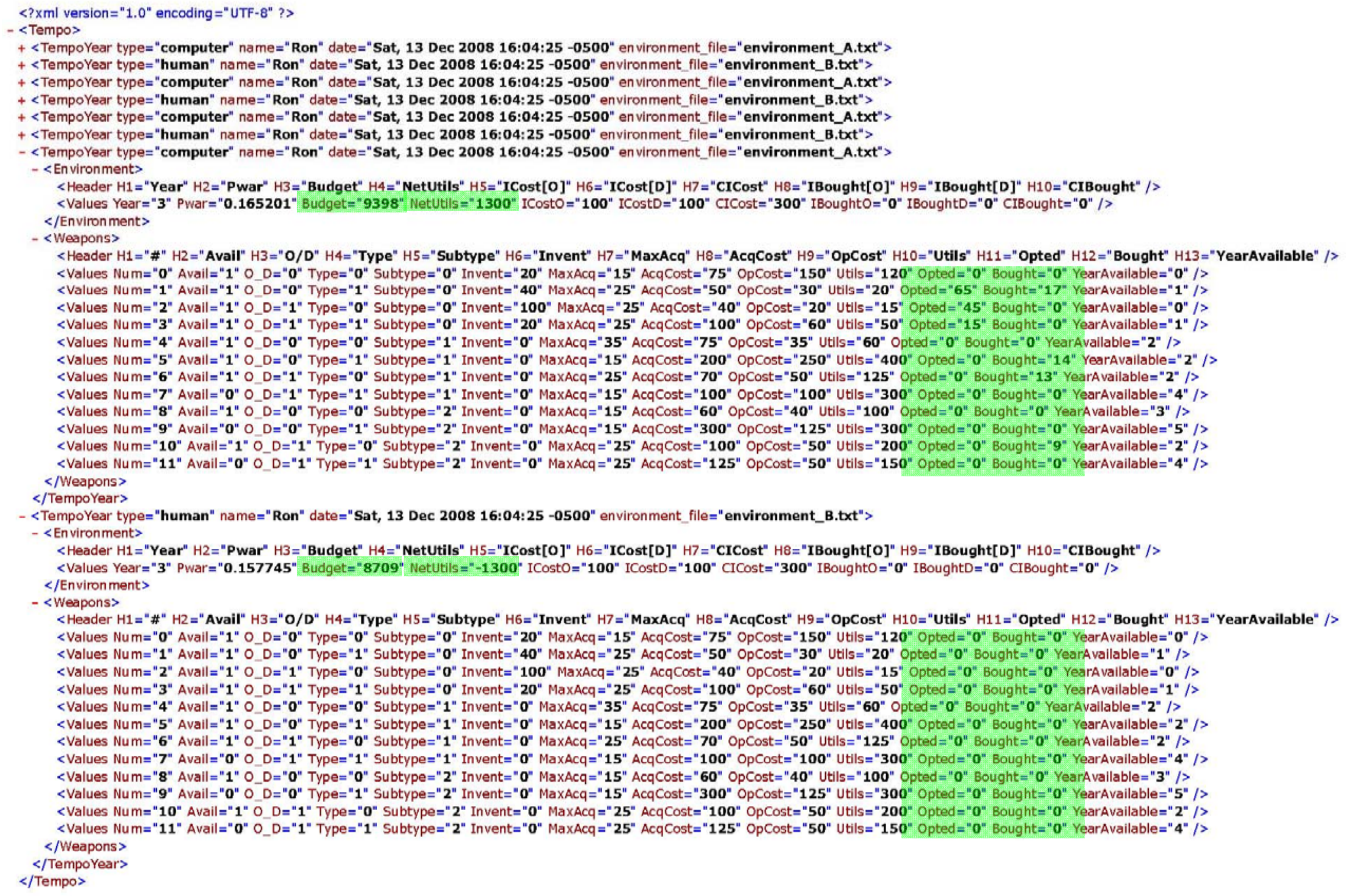
```
  <?xml version="1.0" encoding="UTF-8" ?>
- <Tempo>
  - <TempoYear type="computer" name="Ron" date="Sat, 13 Dec 2008 16:04:25 -0500" environment_file="environment_A.txt">
    - <Environment>
        <Header H1="Year" H2="Pwar" H3="Budget" H4="NetUtils" H5="ICost[O]" H6="ICost[D]" H7="CICost" H8="IBought[O]" H9="IBought[D]" H10="CIBought" />
        <Values Year="0" Pwar="0.1" Budget="8000" NetUtils="0" ICostO="100" ICostD="100" CICost="300" IBoughtO="0" IBoughtD="0" CIBought="0" />
      </Environment>
    - <Weapons>
        <Header H1="#" H2="Avail" H3="O/D" H4="Type" H5="Subtype" H6="Invent" H7="MaxAcq" H8="AcqCost" H9="OpCost" H10="Utils" H11="Opted" H12="Bought" H13="YearAvailable" />
        <Values Num="0" Avail="1" O_D="0" Type="0" Subtype="0" Invent="20" MaxAcq="15" AcqCost="75" OpCost="150" Utils="120" Opted="0" Bought="0" YearAvailable="0" />
        <Values Num="1" Avail="0" O_D="0" Type="1" Subtype="0" Invent="40" MaxAcq="25" AcqCost="50" OpCost="30" Utils="20" Opted="0" Bought="0" YearAvailable="1" />
        <Values Num="2" Avail="1" O_D="1" Type="0" Subtype="0" Invent="100" MaxAcq="25" AcqCost="40" OpCost="20" Utils="15" Opted="0" Bought="0" YearAvailable="0" />
        <Values Num="3" Avail="0" O_D="1" Type="1" Subtype="0" Invent="20" MaxAcq="25" AcqCost="100" OpCost="60" Utils="50" Opted="0" Bought="0" YearAvailable="1" />
        <Values Num="4" Avail="0" O_D="0" Type="0" Subtype="1" Invent="0" MaxAcq="35" AcqCost="75" OpCost="35" Utils="60" Opted="0" Bought="0" YearAvailable="2" />
        <Values Num="5" Avail="0" O_D="0" Type="1" Subtype="1" Invent="0" MaxAcq="15" AcqCost="200" OpCost="250" Utils="400" Opted="0" Bought="0" YearAvailable="2" />
        <Values Num="6" Avail="0" O_D="1" Type="0" Subtype="1" Invent="0" MaxAcq="25" AcqCost="70" OpCost="50" Utils="125" Opted="0" Bought="0" YearAvailable="2" />
        <Values Num="7" Avail="0" O_D="1" Type="1" Subtype="1" Invent="0" MaxAcq="15" AcqCost="100" OpCost="100" Utils="300" Opted="0" Bought="0" YearAvailable="4" />
        <Values Num="8" Avail="0" O_D="0" Type="0" Subtype="2" Invent="0" MaxAcq="15" AcqCost="60" OpCost="40" Utils="100" Opted="0" Bought="0" YearAvailable="3" />
        <Values Num="9" Avail="0" O_D="0" Type="1" Subtype="2" Invent="0" MaxAcq="15" AcqCost="300" OpCost="125" Utils="300" Opted="0" Bought="0" YearAvailable="5" />
        <Values Num="10" Avail="0" O_D="1" Type="0" Subtype="2" Invent="0" MaxAcq="25" AcqCost="100" OpCost="50" Utils="200" Opted="0" Bought="0" YearAvailable="2" />
        <Values Num="11" Avail="0" O_D="1" Type="1" Subtype="2" Invent="0" MaxAcq="25" AcqCost="125" OpCost="50" Utils="150" Opted="0" Bought="0" YearAvailable="4" />
      </Weapons>
    </TempoYear>
  - <TempoYear type="human" name="Ron" date="Sat, 13 Dec 2008 16:04:25 -0500" environment_file="environment_B.txt">
    - <Environment>
        <Header H1="Year" H2="Pwar" H3="Budget" H4="NetUtils" H5="ICost[O]" H6="ICost[D]" H7="CICost" H8="IBought[O]" H9="IBought[D]" H10="CIBought" />
        <Values Year="0" Pwar="0.1" Budget="8000" NetUtils="0" ICostO="100" ICostD="100" CICost="300" IBoughtO="0" IBoughtD="0" CIBought="0" />
      </Environment>
    - <Weapons>
        <Header H1="#" H2="Avail" H3="O/D" H4="Type" H5="Subtype" H6="Invent" H7="MaxAcq" H8="AcqCost" H9="OpCost" H10="Utils" H11="Opted" H12="Bought" H13="YearAvailable" />
        <Values Num="0" Avail="1" O_D="0" Type="0" Subtype="0" Invent="20" MaxAcq="15" AcqCost="75" OpCost="150" Utils="120" Opted="0" Bought="0" YearAvailable="0" />
        <Values Num="1" Avail="0" O_D="0" Type="1" Subtype="0" Invent="40" MaxAcq="25" AcqCost="50" OpCost="30" Utils="20" Opted="0" Bought="0" YearAvailable="1" />
        <Values Num="2" Avail="1" O_D="1" Type="0" Subtype="0" Invent="100" MaxAcq="25" AcqCost="40" OpCost="20" Utils="15" Opted="0" Bought="0" YearAvailable="0" />
        <Values Num="3" Avail="0" O_D="1" Type="1" Subtype="0" Invent="20" MaxAcq="25" AcqCost="100" OpCost="60" Utils="50" Opted="0" Bought="0" YearAvailable="1" />
        <Values Num="4" Avail="0" O_D="0" Type="0" Subtype="1" Invent="0" MaxAcq="35" AcqCost="75" OpCost="35" Utils="60" Opted="0" Bought="0" YearAvailable="2" />
        <Values Num="5" Avail="0" O_D="0" Type="1" Subtype="1" Invent="0" MaxAcq="15" AcqCost="200" OpCost="250" Utils="400" Opted="0" Bought="0" YearAvailable="2" />
        <Values Num="6" Avail="0" O_D="1" Type="0" Subtype="1" Invent="0" MaxAcq="25" AcqCost="70" OpCost="50" Utils="125" Opted="0" Bought="0" YearAvailable="2" />
        <Values Num="7" Avail="0" O_D="1" Type="1" Subtype="1" Invent="0" MaxAcq="15" AcqCost="100" OpCost="100" Utils="300" Opted="0" Bought="0" YearAvailable="4" />
        <Values Num="8" Avail="0" O_D="0" Type="0" Subtype="2" Invent="0" MaxAcq="15" AcqCost="60" OpCost="40" Utils="100" Opted="0" Bought="0" YearAvailable="3" />
        <Values Num="9" Avail="0" O_D="0" Type="1" Subtype="2" Invent="0" MaxAcq="15" AcqCost="300" OpCost="125" Utils="300" Opted="0" Bought="0" YearAvailable="5" />
        <Values Num="10" Avail="0" O_D="1" Type="0" Subtype="2" Invent="0" MaxAcq="25" AcqCost="100" OpCost="50" Utils="200" Opted="0" Bought="0" YearAvailable="2" />
        <Values Num="11" Avail="0" O_D="1" Type="1" Subtype="2" Invent="0" MaxAcq="25" AcqCost="125" OpCost="50" Utils="150" Opted="0" Bought="0" YearAvailable="4" />
      </Weapons>
    </TempoYear>
  + <TempoYear type="computer" name="Ron" date="Sat, 13 Dec 2008 16:04:25 -0500" environment_file="environment_A.txt">
  + <TempoYear type="human" name="Ron" date="Sat, 13 Dec 2008 16:04:25 -0500" environment_file="environment_B.txt">
  + <TempoYear type="computer" name="Ron" date="Sat, 13 Dec 2008 16:04:25 -0500" environment_file="environment_A.txt">
  + <TempoYear type="human" name="Ron" date="Sat, 13 Dec 2008 16:04:25 -0500" environment_file="environment_B.txt">
  + <TempoYear type="computer" name="Ron" date="Sat, 13 Dec 2008 16:04:25 -0500" environment_file="environment_A.txt">
  + <TempoYear type="human" name="Ron" date="Sat, 13 Dec 2008 16:04:25 -0500" environment_file="environment_B.txt">
  </Tempo>
```

Figure 9. TEMPO Version 2 XML Log File–First Year



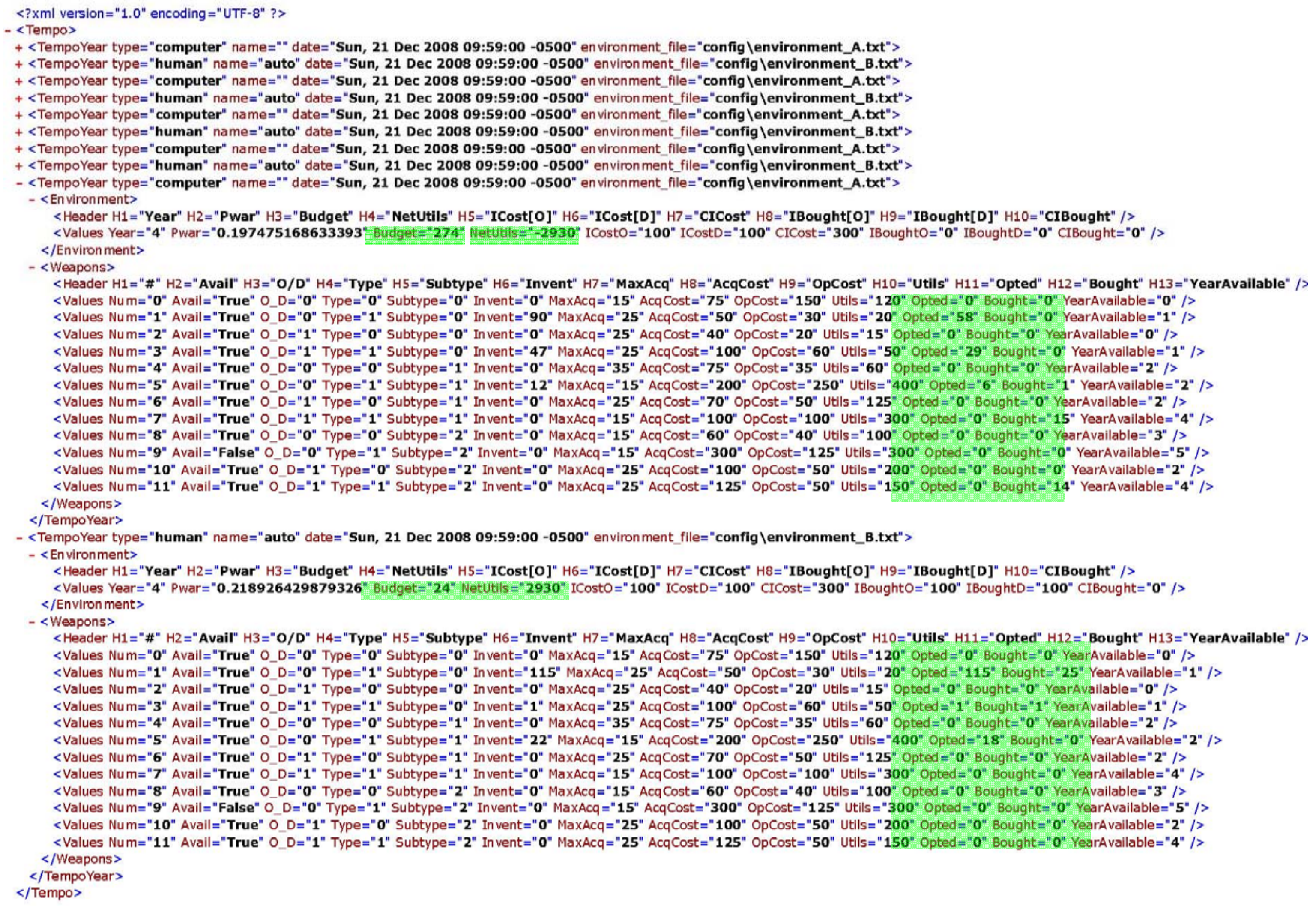
```
<?xml version="1.0" encoding="UTF-8" ?>
- <Tempo>
 - <TempoYear type="computer" name="" date="Sun, 21 Dec 2008 09:59:00 -0500" environment_file="config\environment_A.txt">
  - <Environment>
      <Header H1="Year" H2="Pwar" H3="Budget" H4="NetUtils" H5="ICost[O]" H6="ICost[D]" H7="CICost" H8="IBought[O]" H9="IBought[D]" H10="CIBought" />
      <Values Year="0" Pwar="0.1" Budget="1025" NetUtils="0" ICostO="100" ICostD="100" CICost="300" IBoughtO="0" IBoughtD="0" CIBought="0" />
    </Environment>
  - <Weapons>
      <Header H1="#" H2="Avail" H3="O/D" H4="Type" H5="Subtype" H6="Invent" H7="MaxAcq" H8="AcqCost" H9="OpCost" H10="Utils" H11="Opted" H12="Bought" H13="YearAvailable" />
      <Values Num="0" Avail="True" O_D="0" Type="0" Subtype="0" Invent="20" MaxAcq="15" AcqCost="75" OpCost="150" Utils="120" Opted="20" Bought="13" YearAvailable="0" />
      <Values Num="1" Avail="False" O_D="0" Type="1" Subtype="0" Invent="0" MaxAcq="25" AcqCost="50" OpCost="30" Utils="20" Opted="0" Bought="0" YearAvailable="1" />
      <Values Num="2" Avail="True" O_D="1" Type="0" Subtype="0" Invent="100" MaxAcq="25" AcqCost="40" OpCost="20" Utils="15" Opted="100" Bought="25" YearAvailable="0" />
      <Values Num="3" Avail="False" O_D="1" Type="1" Subtype="0" Invent="0" MaxAcq="25" AcqCost="100" OpCost="60" Utils="50" Opted="0" Bought="0" YearAvailable="1" />
      <Values Num="4" Avail="False" O_D="0" Type="0" Subtype="1" Invent="0" MaxAcq="35" AcqCost="75" OpCost="35" Utils="60" Opted="0" Bought="0" YearAvailable="2" />
      <Values Num="5" Avail="False" O_D="0" Type="1" Subtype="1" Invent="0" MaxAcq="15" AcqCost="200" OpCost="250" Utils="400" Opted="0" Bought="0" YearAvailable="2" />
      <Values Num="6" Avail="False" O_D="1" Type="0" Subtype="1" Invent="0" MaxAcq="25" AcqCost="70" OpCost="50" Utils="125" Opted="0" Bought="0" YearAvailable="2" />
      <Values Num="7" Avail="False" O_D="1" Type="1" Subtype="1" Invent="0" MaxAcq="15" AcqCost="100" OpCost="100" Utils="300" Opted="0" Bought="0" YearAvailable="4" />
      <Values Num="8" Avail="False" O_D="0" Type="0" Subtype="2" Invent="0" MaxAcq="15" AcqCost="60" OpCost="40" Utils="100" Opted="0" Bought="0" YearAvailable="3" />
      <Values Num="9" Avail="False" O_D="0" Type="1" Subtype="2" Invent="0" MaxAcq="15" AcqCost="300" OpCost="125" Utils="300" Opted="0" Bought="0" YearAvailable="5" />
      <Values Num="10" Avail="False" O_D="1" Type="0" Subtype="2" Invent="0" MaxAcq="25" AcqCost="100" OpCost="50" Utils="200" Opted="0" Bought="0" YearAvailable="2" />
      <Values Num="11" Avail="False" O_D="1" Type="1" Subtype="2" Invent="0" MaxAcq="25" AcqCost="125" OpCost="50" Utils="150" Opted="0" Bought="0" YearAvailable="4" />
    </Weapons>
  </TempoYear>
 - <TempoYear type="human" name="auto" date="Sun, 21 Dec 2008 09:59:00 -0500" environment_file="config\environment_B.txt">
  - <Environment>
      <Header H1="Year" H2="Pwar" H3="Budget" H4="NetUtils" H5="ICost[O]" H6="ICost[D]" H7="CICost" H8="IBought[O]" H9="IBought[D]" H10="CIBought" />
      <Values Year="0" Pwar="0.1" Budget="675" NetUtils="0" ICostO="100" ICostD="100" CICost="300" IBoughtO="100" IBoughtD="100" CIBought="0" />
    </Environment>
  - <Weapons>
      <Header H1="#" H2="Avail" H3="O/D" H4="Type" H5="Subtype" H6="Invent" H7="MaxAcq" H8="AcqCost" H9="OpCost" H10="Utils" H11="Opted" H12="Bought" H13="YearAvailable" />
      <Values Num="0" Avail="True" O_D="0" Type="0" Subtype="0" Invent="20" MaxAcq="15" AcqCost="75" OpCost="150" Utils="120" Opted="20" Bought="15" YearAvailable="0" />
      <Values Num="1" Avail="False" O_D="0" Type="1" Subtype="0" Invent="0" MaxAcq="25" AcqCost="50" OpCost="30" Utils="20" Opted="0" Bought="0" YearAvailable="1" />
      <Values Num="2" Avail="True" O_D="1" Type="0" Subtype="0" Invent="100" MaxAcq="25" AcqCost="40" OpCost="20" Utils="15" Opted="100" Bought="25" YearAvailable="0" />
      <Values Num="3" Avail="False" O_D="1" Type="1" Subtype="0" Invent="0" MaxAcq="25" AcqCost="100" OpCost="60" Utils="50" Opted="0" Bought="0" YearAvailable="1" />
      <Values Num="4" Avail="False" O_D="0" Type="0" Subtype="1" Invent="0" MaxAcq="35" AcqCost="75" OpCost="35" Utils="60" Opted="0" Bought="0" YearAvailable="2" />
      <Values Num="5" Avail="False" O_D="0" Type="1" Subtype="1" Invent="0" MaxAcq="15" AcqCost="200" OpCost="250" Utils="400" Opted="0" Bought="0" YearAvailable="2" />
      <Values Num="6" Avail="False" O_D="1" Type="0" Subtype="1" Invent="0" MaxAcq="25" AcqCost="70" OpCost="50" Utils="125" Opted="0" Bought="0" YearAvailable="2" />
      <Values Num="7" Avail="False" O_D="1" Type="1" Subtype="1" Invent="0" MaxAcq="15" AcqCost="100" OpCost="100" Utils="300" Opted="0" Bought="0" YearAvailable="4" />
      <Values Num="8" Avail="False" O_D="0" Type="0" Subtype="2" Invent="0" MaxAcq="15" AcqCost="60" OpCost="40" Utils="100" Opted="0" Bought="0" YearAvailable="3" />
      <Values Num="9" Avail="False" O_D="0" Type="1" Subtype="2" Invent="0" MaxAcq="15" AcqCost="300" OpCost="125" Utils="300" Opted="0" Bought="0" YearAvailable="5" />
      <Values Num="10" Avail="False" O_D="1" Type="0" Subtype="2" Invent="0" MaxAcq="25" AcqCost="100" OpCost="50" Utils="200" Opted="0" Bought="0" YearAvailable="2" />
      <Values Num="11" Avail="False" O_D="1" Type="1" Subtype="2" Invent="0" MaxAcq="25" AcqCost="125" OpCost="50" Utils="150" Opted="0" Bought="0" YearAvailable="4" />
    </Weapons>
  </TempoYear>
 + <TempoYear type="computer" name="" date="Sun, 21 Dec 2008 09:59:00 -0500" environment_file="config\environment_A.txt">
 + <TempoYear type="human" name="auto" date="Sun, 21 Dec 2008 09:59:00 -0500" environment_file="config\environment_B.txt">
 + <TempoYear type="computer" name="" date="Sun, 21 Dec 2008 09:59:00 -0500" environment_file="config\environment_A.txt">
 + <TempoYear type="human" name="auto" date="Sun, 21 Dec 2008 09:59:00 -0500" environment_file="config\environment_B.txt">
 + <TempoYear type="computer" name="" date="Sun, 21 Dec 2008 09:59:00 -0500" environment_file="config\environment_A.txt">
 + <TempoYear type="human" name="auto" date="Sun, 21 Dec 2008 09:59:00 -0500" environment_file="config\environment_B.txt">
 + <TempoYear type="computer" name="" date="Sun, 21 Dec 2008 09:59:00 -0500" environment_file="config\environment_A.txt">
 + <TempoYear type="human" name="auto" date="Sun, 21 Dec 2008 09:59:00 -0500" environment_file="config\environment_B.txt">
</Tempo>
```

Figure 10. TEMPO Version 3 XML Log File–First Year



```
  <?xml version="1.0" encoding="UTF-8" ?>
- <Tempo>
  + <TempoYear type="computer" name="Ron" date="Sat, 13 Dec 2008 16:04:25 -0500" environment_file="environment_A.txt">
  + <TempoYear type="human" name="Ron" date="Sat, 13 Dec 2008 16:04:25 -0500" environment_file="environment_B.txt">
  + <TempoYear type="computer" name="Ron" date="Sat, 13 Dec 2008 16:04:25 -0500" environment_file="environment_A.txt">
  + <TempoYear type="human" name="Ron" date="Sat, 13 Dec 2008 16:04:25 -0500" environment_file="environment_B.txt">
  + <TempoYear type="computer" name="Ron" date="Sat, 13 Dec 2008 16:04:25 -0500" environment_file="environment_A.txt">
  + <TempoYear type="human" name="Ron" date="Sat, 13 Dec 2008 16:04:25 -0500" environment_file="environment_B.txt">
  - <TempoYear type="computer" name="Ron" date="Sat, 13 Dec 2008 16:04:25 -0500" environment_file="environment_A.txt">
    - <Environment>
        <Header H1="Year" H2="Pwar" H3="Budget" H4="NetUtils" H5="ICost[O]" H6="ICost[D]" H7="CICost" H8="IBought[O]" H9="IBought[D]" H10="CIBought" />
        <Values Year="3" Pwar="0.165201" Budget="9398" NetUtils="1300" ICostO="100" ICostD="100" CICost="300" IBoughtO="0" IBoughtD="0" CIBought="0" />
      </Environment>
    - <Weapons>
        <Header H1="#" H2="Avail" H3="O/D" H4="Type" H5="Subtype" H6="Invent" H7="MaxAcq" H8="AcqCost" H9="OpCost" H10="Utils" H11="Opted" H12="Bought" H13="YearAvailable" />
        <Values Num="0" Avail="1" O_D="0" Type="0" Subtype="0" Invent="20" MaxAcq="15" AcqCost="75" OpCost="150" Utils="120" Opted="0" Bought="0" YearAvailable="0" />
        <Values Num="1" Avail="1" O_D="0" Type="1" Subtype="0" Invent="40" MaxAcq="25" AcqCost="50" OpCost="30" Utils="20" Opted="65" Bought="17" YearAvailable="1" />
        <Values Num="2" Avail="1" O_D="1" Type="0" Subtype="0" Invent="100" MaxAcq="25" AcqCost="40" OpCost="20" Utils="15" Opted="45" Bought="0" YearAvailable="0" />
        <Values Num="3" Avail="1" O_D="1" Type="1" Subtype="0" Invent="20" MaxAcq="25" AcqCost="100" OpCost="60" Utils="50" Opted="15" Bought="0" YearAvailable="1" />
        <Values Num="4" Avail="1" O_D="0" Type="0" Subtype="1" Invent="0" MaxAcq="35" AcqCost="75" OpCost="35" Utils="60" Opted="0" Bought="0" YearAvailable="2" />
        <Values Num="5" Avail="1" O_D="0" Type="1" Subtype="1" Invent="0" MaxAcq="15" AcqCost="200" OpCost="250" Utils="400" Opted="0" Bought="14" YearAvailable="2" />
        <Values Num="6" Avail="1" O_D="1" Type="0" Subtype="1" Invent="0" MaxAcq="25" AcqCost="70" OpCost="50" Utils="125" Opted="0" Bought="13" YearAvailable="2" />
        <Values Num="7" Avail="0" O_D="1" Type="1" Subtype="1" Invent="0" MaxAcq="15" AcqCost="100" OpCost="100" Utils="300" Opted="0" Bought="0" YearAvailable="4" />
        <Values Num="8" Avail="1" O_D="0" Type="0" Subtype="2" Invent="0" MaxAcq="15" AcqCost="60" OpCost="40" Utils="100" Opted="0" Bought="0" YearAvailable="3" />
        <Values Num="9" Avail="0" O_D="0" Type="1" Subtype="2" Invent="0" MaxAcq="15" AcqCost="300" OpCost="125" Utils="300" Opted="0" Bought="0" YearAvailable="5" />
        <Values Num="10" Avail="1" O_D="1" Type="0" Subtype="2" Invent="0" MaxAcq="25" AcqCost="100" OpCost="50" Utils="200" Opted="0" Bought="9" YearAvailable="2" />
        <Values Num="11" Avail="0" O_D="1" Type="1" Subtype="2" Invent="0" MaxAcq="25" AcqCost="125" OpCost="50" Utils="150" Opted="0" Bought="0" YearAvailable="4" />
      </Weapons>
    </TempoYear>
  - <TempoYear type="human" name="Ron" date="Sat, 13 Dec 2008 16:04:25 -0500" environment_file="environment_B.txt">
    - <Environment>
        <Header H1="Year" H2="Pwar" H3="Budget" H4="NetUtils" H5="ICost[O]" H6="ICost[D]" H7="CICost" H8="IBought[O]" H9="IBought[D]" H10="CIBought" />
        <Values Year="3" Pwar="0.157745" Budget="8709" NetUtils="-1300" ICostO="100" ICostD="100" CICost="300" IBoughtO="0" IBoughtD="0" CIBought="0" />
      </Environment>
    - <Weapons>
        <Header H1="#" H2="Avail" H3="O/D" H4="Type" H5="Subtype" H6="Invent" H7="MaxAcq" H8="AcqCost" H9="OpCost" H10="Utils" H11="Opted" H12="Bought" H13="YearAvailable" />
        <Values Num="0" Avail="1" O_D="0" Type="0" Subtype="0" Invent="20" MaxAcq="15" AcqCost="75" OpCost="150" Utils="120" Opted="0" Bought="0" YearAvailable="0" />
        <Values Num="1" Avail="1" O_D="0" Type="1" Subtype="0" Invent="40" MaxAcq="25" AcqCost="50" OpCost="30" Utils="20" Opted="0" Bought="0" YearAvailable="1" />
        <Values Num="2" Avail="1" O_D="1" Type="0" Subtype="0" Invent="100" MaxAcq="25" AcqCost="40" OpCost="20" Utils="15" Opted="0" Bought="0" YearAvailable="0" />
        <Values Num="3" Avail="1" O_D="1" Type="1" Subtype="0" Invent="20" MaxAcq="25" AcqCost="100" OpCost="60" Utils="50" Opted="0" Bought="0" YearAvailable="1" />
        <Values Num="4" Avail="1" O_D="0" Type="0" Subtype="1" Invent="0" MaxAcq="35" AcqCost="75" OpCost="35" Utils="60" Opted="0" Bought="0" YearAvailable="2" />
        <Values Num="5" Avail="1" O_D="0" Type="1" Subtype="1" Invent="0" MaxAcq="15" AcqCost="200" OpCost="250" Utils="400" Opted="0" Bought="0" YearAvailable="2" />
        <Values Num="6" Avail="1" O_D="1" Type="0" Subtype="1" Invent="0" MaxAcq="25" AcqCost="70" OpCost="50" Utils="125" Opted="0" Bought="0" YearAvailable="2" />
        <Values Num="7" Avail="0" O_D="1" Type="1" Subtype="1" Invent="0" MaxAcq="15" AcqCost="100" OpCost="100" Utils="300" Opted="0" Bought="0" YearAvailable="4" />
        <Values Num="8" Avail="1" O_D="0" Type="0" Subtype="2" Invent="0" MaxAcq="15" AcqCost="60" OpCost="40" Utils="100" Opted="0" Bought="0" YearAvailable="3" />
        <Values Num="9" Avail="0" O_D="0" Type="1" Subtype="2" Invent="0" MaxAcq="15" AcqCost="300" OpCost="125" Utils="300" Opted="0" Bought="0" YearAvailable="5" />
        <Values Num="10" Avail="1" O_D="1" Type="0" Subtype="2" Invent="0" MaxAcq="25" AcqCost="100" OpCost="50" Utils="200" Opted="0" Bought="0" YearAvailable="2" />
        <Values Num="11" Avail="0" O_D="1" Type="1" Subtype="2" Invent="0" MaxAcq="25" AcqCost="125" OpCost="50" Utils="150" Opted="0" Bought="0" YearAvailable="4" />
      </Weapons>
    </TempoYear>
  </Tempo>
```

Figure 11. TEMPO Version 2 XML Log File–Last Year



```
<?xml version="1.0" encoding="UTF-8" ?>
- <Tempo>
  + <TempoYear type="computer" name="" date="Sun, 21 Dec 2008 09:59:00 -0500" environment_file="config\environment_A.txt">
  + <TempoYear type="human" name="auto" date="Sun, 21 Dec 2008 09:59:00 -0500" environment_file="config\environment_B.txt">
  + <TempoYear type="computer" name="" date="Sun, 21 Dec 2008 09:59:00 -0500" environment_file="config\environment_A.txt">
  + <TempoYear type="human" name="auto" date="Sun, 21 Dec 2008 09:59:00 -0500" environment_file="config\environment_B.txt">
  + <TempoYear type="computer" name="" date="Sun, 21 Dec 2008 09:59:00 -0500" environment_file="config\environment_A.txt">
  + <TempoYear type="human" name="auto" date="Sun, 21 Dec 2008 09:59:00 -0500" environment_file="config\environment_B.txt">
  + <TempoYear type="computer" name="" date="Sun, 21 Dec 2008 09:59:00 -0500" environment_file="config\environment_A.txt">
  + <TempoYear type="human" name="auto" date="Sun, 21 Dec 2008 09:59:00 -0500" environment_file="config\environment_B.txt">
  - <TempoYear type="computer" name="" date="Sun, 21 Dec 2008 09:59:00 -0500" environment_file="config\environment_A.txt">
    - <Environment>
        <Header H1="Year" H2="Pwar" H3="Budget" H4="NetUtils" H5="ICost[O]" H6="ICost[D]" H7="CICost" H8="IBought[O]" H9="IBought[D]" H10="CIBought" />
        <Values Year="4" Pwar="0.197475168633393" Budget="274" NetUtils="-2930" ICostO="100" ICostD="100" CICost="300" IBoughtO="0" IBoughtD="0" CIBought="0" />
      </Environment>
    - <Weapons>
        <Header H1="#" H2="Avail" H3="O/D" H4="Type" H5="Subtype" H6="Invent" H7="MaxAcq" H8="AcqCost" H9="OpCost" H10="Utils" H11="Opted" H12="Bought" H13="YearAvailable" />
        <Values Num="0" Avail="True" O_D="0" Type="0" Subtype="0" Invent="0" MaxAcq="15" AcqCost="75" OpCost="150" Utils="120" Opted="0" Bought="0" YearAvailable="0" />
        <Values Num="1" Avail="True" O_D="0" Type="1" Subtype="0" Invent="90" MaxAcq="25" AcqCost="50" OpCost="30" Utils="20" Opted="58" Bought="0" YearAvailable="1" />
        <Values Num="2" Avail="True" O_D="1" Type="0" Subtype="0" Invent="0" MaxAcq="25" AcqCost="40" OpCost="20" Utils="15" Opted="0" Bought="0" YearAvailable="0" />
        <Values Num="3" Avail="True" O_D="1" Type="1" Subtype="0" Invent="47" MaxAcq="25" AcqCost="100" OpCost="60" Utils="50" Opted="29" Bought="0" YearAvailable="1" />
        <Values Num="4" Avail="True" O_D="0" Type="0" Subtype="1" Invent="0" MaxAcq="35" AcqCost="75" OpCost="35" Utils="60" Opted="0" Bought="0" YearAvailable="2" />
        <Values Num="5" Avail="True" O_D="0" Type="1" Subtype="1" Invent="12" MaxAcq="15" AcqCost="200" OpCost="250" Utils="400" Opted="6" Bought="1" YearAvailable="2" />
        <Values Num="6" Avail="True" O_D="1" Type="0" Subtype="1" Invent="0" MaxAcq="25" AcqCost="70" OpCost="50" Utils="125" Opted="0" Bought="0" YearAvailable="2" />
        <Values Num="7" Avail="True" O_D="1" Type="1" Subtype="1" Invent="0" MaxAcq="15" AcqCost="100" OpCost="100" Utils="300" Opted="0" Bought="15" YearAvailable="4" />
        <Values Num="8" Avail="True" O_D="0" Type="0" Subtype="2" Invent="0" MaxAcq="15" AcqCost="60" OpCost="40" Utils="100" Opted="0" Bought="0" YearAvailable="3" />
        <Values Num="9" Avail="False" O_D="0" Type="1" Subtype="2" Invent="0" MaxAcq="15" AcqCost="300" OpCost="125" Utils="300" Opted="0" Bought="0" YearAvailable="5" />
        <Values Num="10" Avail="True" O_D="1" Type="0" Subtype="2" Invent="0" MaxAcq="25" AcqCost="100" OpCost="50" Utils="200" Opted="0" Bought="0" YearAvailable="2" />
        <Values Num="11" Avail="True" O_D="1" Type="1" Subtype="2" Invent="0" MaxAcq="25" AcqCost="125" OpCost="50" Utils="150" Opted="0" Bought="14" YearAvailable="4" />
      </Weapons>
    </TempoYear>
  - <TempoYear type="human" name="auto" date="Sun, 21 Dec 2008 09:59:00 -0500" environment_file="config\environment_B.txt">
    - <Environment>
        <Header H1="Year" H2="Pwar" H3="Budget" H4="NetUtils" H5="ICost[O]" H6="ICost[D]" H7="CICost" H8="IBought[O]" H9="IBought[D]" H10="CIBought" />
        <Values Year="4" Pwar="0.218926429879326" Budget="24" NetUtils="2930" ICostO="100" ICostD="100" CICost="300" IBoughtO="100" IBoughtD="100" CIBought="0" />
      </Environment>
    - <Weapons>
        <Header H1="#" H2="Avail" H3="O/D" H4="Type" H5="Subtype" H6="Invent" H7="MaxAcq" H8="AcqCost" H9="OpCost" H10="Utils" H11="Opted" H12="Bought" H13="YearAvailable" />
        <Values Num="0" Avail="True" O_D="0" Type="0" Subtype="0" Invent="0" MaxAcq="15" AcqCost="75" OpCost="150" Utils="120" Opted="0" Bought="0" YearAvailable="0" />
        <Values Num="1" Avail="True" O_D="0" Type="1" Subtype="0" Invent="115" MaxAcq="25" AcqCost="50" OpCost="30" Utils="20" Opted="115" Bought="25" YearAvailable="1" />
        <Values Num="2" Avail="True" O_D="1" Type="0" Subtype="0" Invent="0" MaxAcq="25" AcqCost="40" OpCost="20" Utils="15" Opted="0" Bought="0" YearAvailable="0" />
        <Values Num="3" Avail="True" O_D="1" Type="1" Subtype="0" Invent="1" MaxAcq="25" AcqCost="100" OpCost="60" Utils="50" Opted="1" Bought="1" YearAvailable="1" />
        <Values Num="4" Avail="True" O_D="0" Type="0" Subtype="1" Invent="0" MaxAcq="35" AcqCost="75" OpCost="35" Utils="60" Opted="0" Bought="0" YearAvailable="2" />
        <Values Num="5" Avail="True" O_D="0" Type="1" Subtype="1" Invent="22" MaxAcq="15" AcqCost="200" OpCost="250" Utils="400" Opted="18" Bought="0" YearAvailable="2" />
        <Values Num="6" Avail="True" O_D="1" Type="0" Subtype="1" Invent="0" MaxAcq="25" AcqCost="70" OpCost="50" Utils="125" Opted="0" Bought="0" YearAvailable="2" />
        <Values Num="7" Avail="True" O_D="1" Type="1" Subtype="1" Invent="0" MaxAcq="15" AcqCost="100" OpCost="100" Utils="300" Opted="0" Bought="0" YearAvailable="4" />
        <Values Num="8" Avail="True" O_D="0" Type="0" Subtype="2" Invent="0" MaxAcq="15" AcqCost="60" OpCost="40" Utils="100" Opted="0" Bought="0" YearAvailable="3" />
        <Values Num="9" Avail="False" O_D="0" Type="1" Subtype="2" Invent="0" MaxAcq="15" AcqCost="300" OpCost="125" Utils="300" Opted="0" Bought="0" YearAvailable="5" />
        <Values Num="10" Avail="True" O_D="1" Type="0" Subtype="2" Invent="0" MaxAcq="25" AcqCost="100" OpCost="50" Utils="200" Opted="0" Bought="0" YearAvailable="2" />
        <Values Num="11" Avail="True" O_D="1" Type="1" Subtype="2" Invent="0" MaxAcq="25" AcqCost="125" OpCost="50" Utils="150" Opted="0" Bought="0" YearAvailable="4" />
      </Weapons>
    </TempoYear>
</Tempo>
```

Figure 12. TEMPO Version 3 XML Log File–Last Year

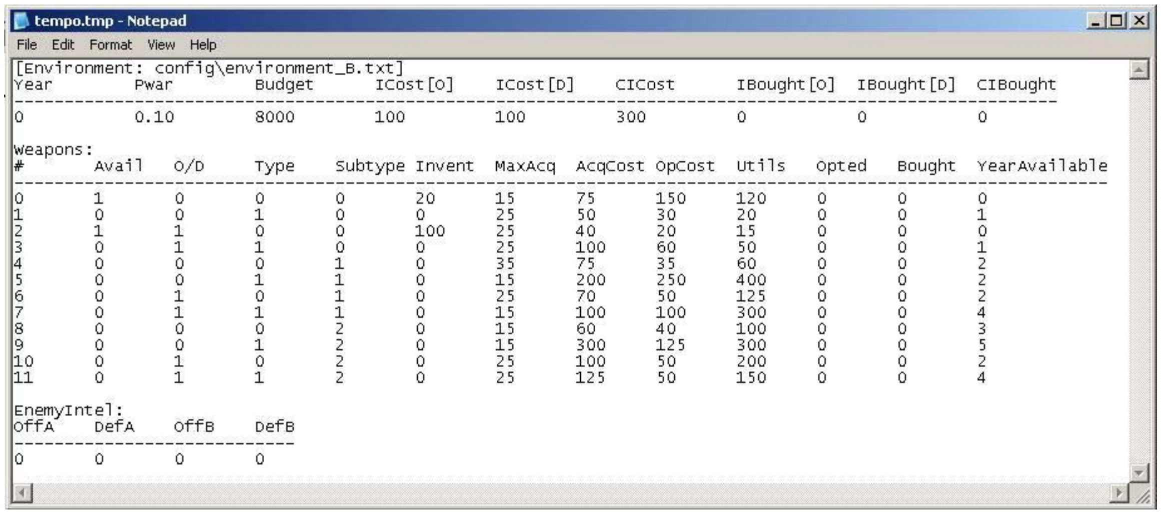
This second feature required a major update to the existing TEMPO.ATL. Previously, an automatic play feature was unnecessary. In order to execute automatic strategies, TEMPO.ATL required:

1. Capture of the current year's environment
2. Capture of the current year's weapon availabilities
3. Capture of intelligence data, if purchased
4. An algorithm to analyze the captured data
5. An algorithm to provide the weapon operation list and weapon purchase list
6. A method to commit the weapon operation and purchase lists

Items 1–3 were implemented by developing a turn-by-turn 'interface file.' Items 4 and 5 were implemented by executing the specific rule-based strategies (discussed in the 'MATLAB Strategy Package Development' section). Item 6 was implemented by designing TEMPO to read the interface file data and executing the weapon allocation decisions automatically. Visual Basic.NET was selected as the desired development environment for the TEMPO update (hence TEMPO.NET) because user interfaces can be quickly prototyped and it is a versatile, yet simple language to use. MATLAB was used to execute the rule-based strategies since it is a software language specifically designed to perform mathematical functions. However, using two different software development environments presented some unique challenges.

B. DEVELOPMENT CHALLENGES

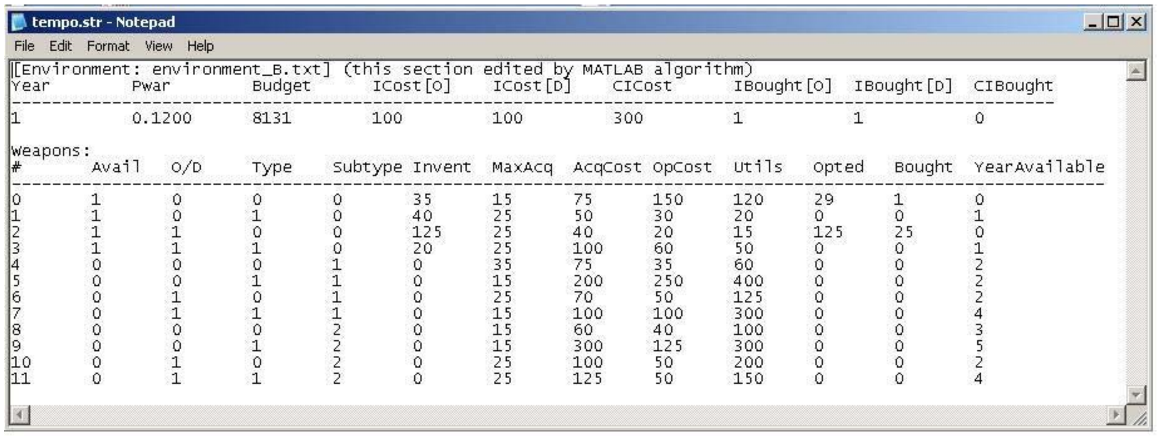
Interfacing data between MATLAB and Visual Basic.NET development environments presented the most difficult challenge. Due to a lack of programming experience, the simplest method—transferring weapon data through reading and writing of plain text files—was chosen. For any given turn (i.e., year of play) the overall approach is the same. First, TEMPO.NET creates a plain text file—tempo.ini'—listing which strategy the user requires. This is a simple text file that the MATLAB executable application eventually requires to execute the appropriate strategy. Next, TEMPO.NET creates a plain text interface file—‘tempo.tmp’—with the year’s environment and weapon availability data, and intelligence purchased, if applicable. TEMPO.NET then calls a previously compiled MATLAB standalone executable program (‘runStrategyExe.exe’) to read the data within these two plain text files, execute the strategy, and output the strategy results into a similar plain text file—‘tempo.str.’ Figure 13 displays a sample ‘tempo.tmp’ file from TEMPO.NET read by MATLAB.



```
[Environment: config\environment_B.txt]
Year        Pwar        Budget      ICost[O]    ICost[D]    CICost      IBought[O]  IBought[D]  CIBought
--------------------------------------------------------------------------------------------------------
0           0.10        8000        100         100         300         0           0           0

Weapons:
#       Avail   O/D     Type    Subtype Invent  MaxAcq  AcqCost OpCost  Utils   Opted   Bought  YearAvailable
-------------------------------------------------------------------------------------------------------------
0       1       0       0       0       20      15      75      150     120     0       0       0
1       0       0       1       0       0       25      50      30      20      0       0       1
2       1       1       0       0       100     25      40      20      15      0       0       0
3       0       1       1       0       0       25      100     60      50      0       0       1
4       0       0       0       1       0       35      75      35      60      0       0       2
5       0       0       1       1       0       15      200     250     400     0       0       2
6       0       1       0       1       0       25      70      50      125     0       0       2
7       0       1       1       1       0       15      100     100     300     0       0       4
8       0       0       0       2       0       15      60      40      100     0       0       3
9       0       0       1       2       0       15      300     125     300     0       0       5
10      0       1       0       2       0       25      100     50      200     0       0       2
11      0       1       1       2       0       25      125     50      150     0       0       4

EnemyIntel:
OffA    DefA    OffB    DefB
-----------------------------
0       0       0       0
```

Figure 13. 'Tempo.tmp'

‘runStrategyExe.exe’ executes the strategy and writes a new text file (‘tempo.str’) to the same directory with the appropriate ‘operate’ and ‘buy’ lists. Figure 14 displays the MATLAB-modified input text file, ‘tempo.str’.



```
[Environment: environment_B.txt] (this section edited by MATLAB algorithm)
Year        Pwar        Budget      ICost[O]    ICost[D]    CICost      IBought[O]  IBought[D]  CIBought
--------------------------------------------------------------------------------------------------------
1           0.1200      8131        100         100         300         1           1           0

Weapons:
#       Avail   O/D     Type    Subtype Invent  MaxAcq  AcqCost OpCost  Utils   Opted   Bought  YearAvailable
-------------------------------------------------------------------------------------------------------------
0       1       0       0       0       35      15      75      150     120     29      1       0
1       1       0       1       0       40      25      50      30      20      0       0       1
2       1       1       0       0       125     25      40      20      15      125     25      0
3       1       1       1       0       20      25      100     60      50      0       0       1
4       0       0       0       1       0       35      75      35      60      0       0       2
5       0       0       1       1       0       15      200     250     400     0       0       2
6       0       1       0       1       0       25      70      50      125     0       0       2
7       0       1       1       1       0       15      100     100     300     0       0       4
8       0       0       0       2       0       15      60      40      100     0       0       3
9       0       0       1       2       0       15      300     125     300     0       0       5
10      0       1       0       2       0       25      100     50      200     0       0       2
11      0       1       1       2       0       25      125     50      150     0       0       4
```

Figure 14. 'Tempo.str'

Once complete, TEMPO.NET then imports this information back into the appropriate text boxes and the commits the weapon and intelligence allocation. This process repeats for every year until war occurs. The most difficult challenge involved timing TEMPO.NET to allow time for the MATLAB strategy to execute without overwriting files in the process. In order to do so, multi-thread execution was added to TEMPO.NET. One thread executes game play while the other continuously monitors for the ‘input file.’ Due to this implementation, the execution of the game in automatic mode updates the user interface unpredictably, so it is difficult to watch a game being played. Although not ideal, this solution provided the necessary functionality to support a consistent information exchange between TEMPO.NET and the MATLAB executable program. A more elegant solution would have been to compile a Dynamic Link Library (DLL) containing all the MATLAB functions for Visual Basic to call. This could also improve performance, as the current solution takes a considerable amount of time to execute (approximately 7 seconds per turn). Unfortunately, lack of programming experience rendered the DLL option too time consuming.

The second major challenge was verifying the MATLAB strategy execution performance. Verification was required in order to ensure proper results from the automated trials were captured. However, many possible situations could occur during each turn. The verification approach was to categorize all possible situations into a test case category. A manually edited text file was produced to represent each test case. The verification of a strategy is successful once it passes all test case categories. These test case categories are described in Section 0.

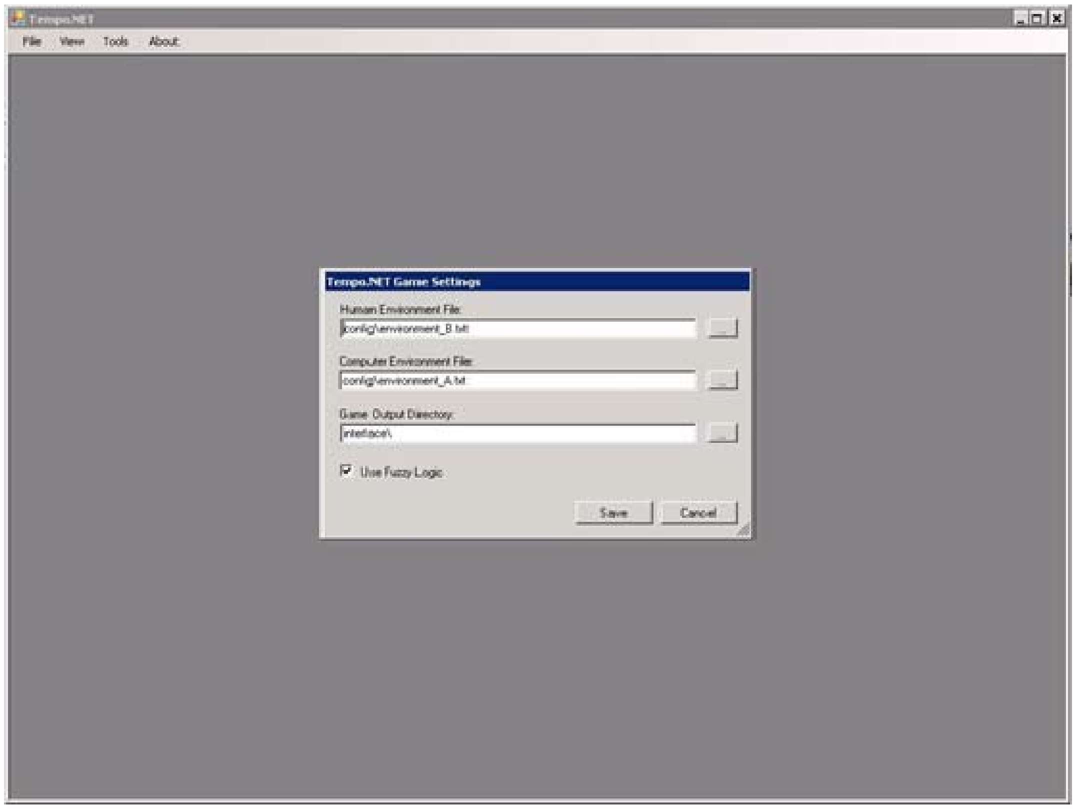
Ultimately, integrating SEVER and TEMPO is the objective. Therefore, SEVER requirements were also considered during all model and strategy software updates and development. The rest of this chapter describes SEVER, TEMPO, an example implementation of SEVER, and preliminary application of SEVER to TEMPO, to enhance the reader’s understanding of the SEVER algorithm and its application.

C. TEMPO.NET DEVELOPMENT

These considerations constitute the basis for this publication. They are entrance criteria (essentially, requirements) that must be met prior to embarking on SEVER's software development.

- TEMPO—Include method to perform automated data entry.
- TEMPO—Speed up player data input for more efficient human trials
- TEMPO—Capture turn-by-turn data to evaluate real-time decision impacts (necessary for future SEVER implementation)
- MATLAB—Analyze large numbers of game performance logs by decomposing performance measures to present those measures statistically based on the implemented strategy.

Figures 15 through 22 display the various features of TEMPO Version 3. These figures are intended to be self-explanatory.



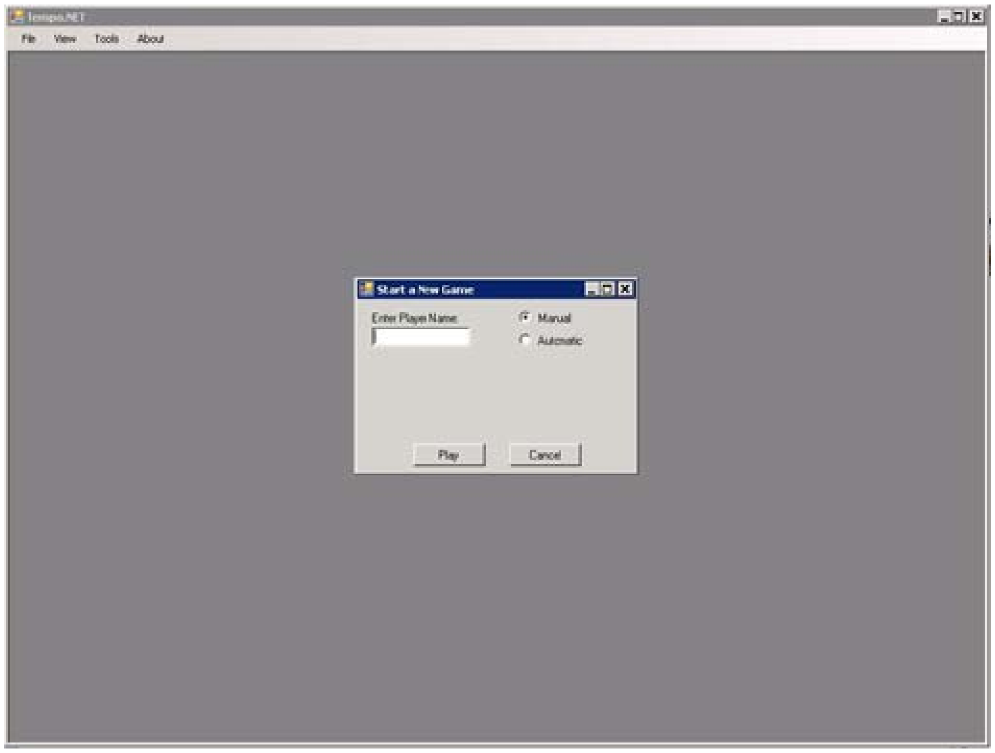
Human Environment File: config\environment_B.txt

Computer Environment File: config\environment_A.txt

Game Output Directory: interface\

[✓] Use Fuzzy Logic

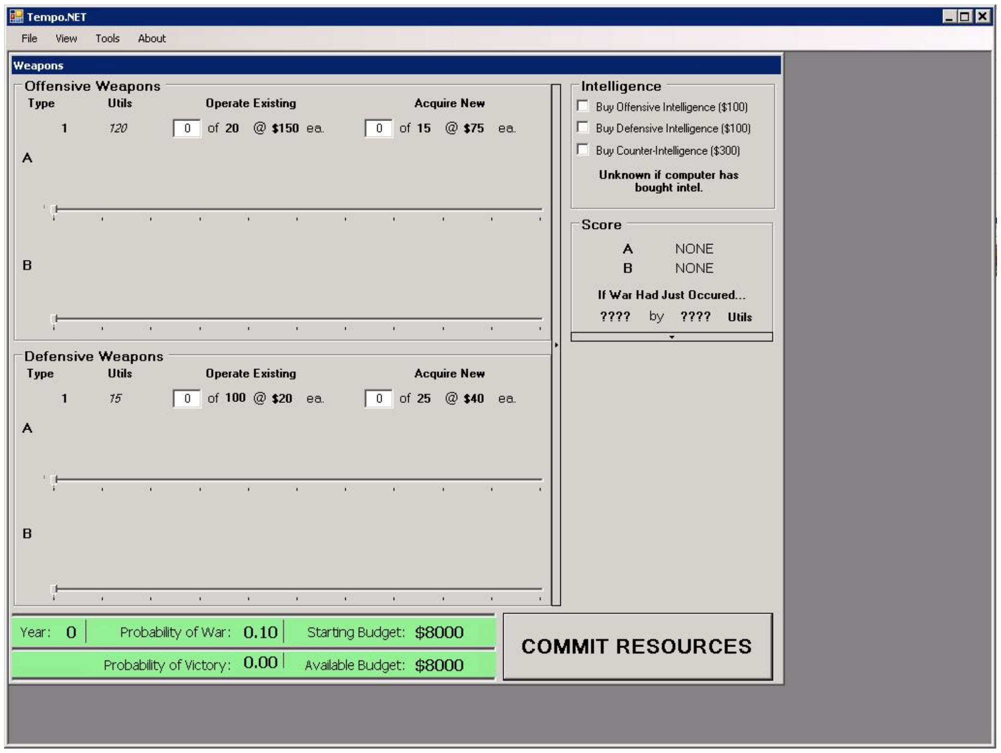
Figure 15. TEMPO Version 3 Configuration Setup



Enter Player Name:

[✓ Manual] Automatic

Figure 16. TEMPO Version 3 Startup Dialog Box



Offensive Weapons

Type	Utils	Operate Existing	Acquire New
1	120	0 of 20 @ $150 ea.	0 of 15 @ $75 ea.

A

B

Defensive Weapons

Type	Utils	Operate Existing	Acquire New
1	15	0 of 100 @ $20 ea.	0 of 25 @ $40 ea.

A

B

Intelligence

- [ ] Buy Offensive Intelligence ($100)
- [ ] Buy Defensive Intelligence ($100)
- [ ] Buy Counter-Intelligence ($300)

Unknown if computer has bought intel.

Score

A	NONE
B	NONE

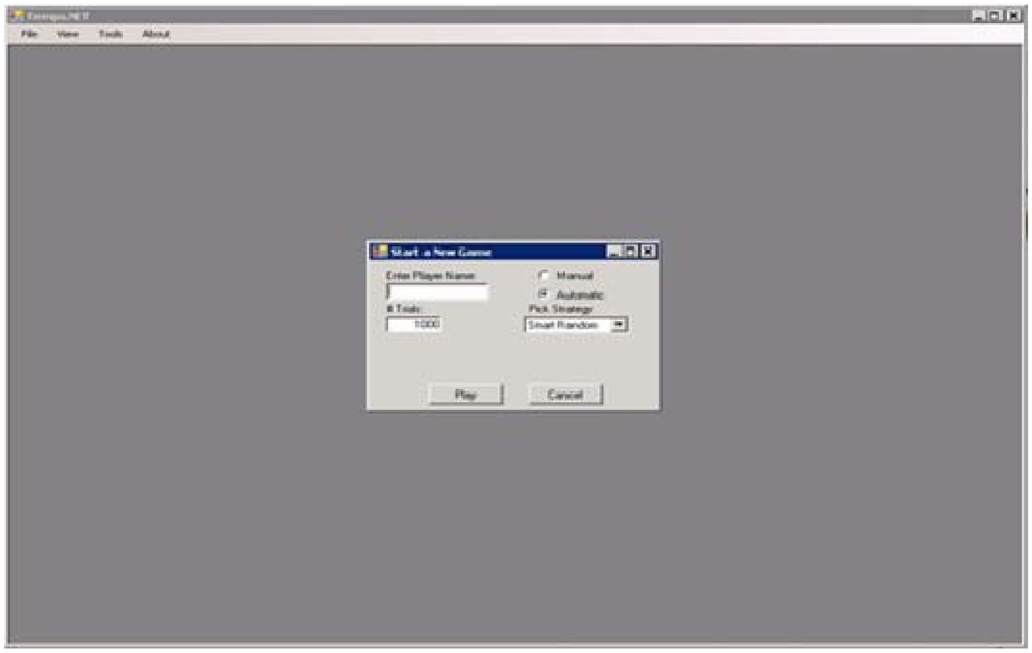
If War Had Just Occured...

???? by ???? Utils

Year: 0 | Probability of War: 0.10 | Starting Budget: $8000

Probability of Victory: 0.00 | Available Budget: $8000

Figure 17. TEMPO Version 3 Main User Interface



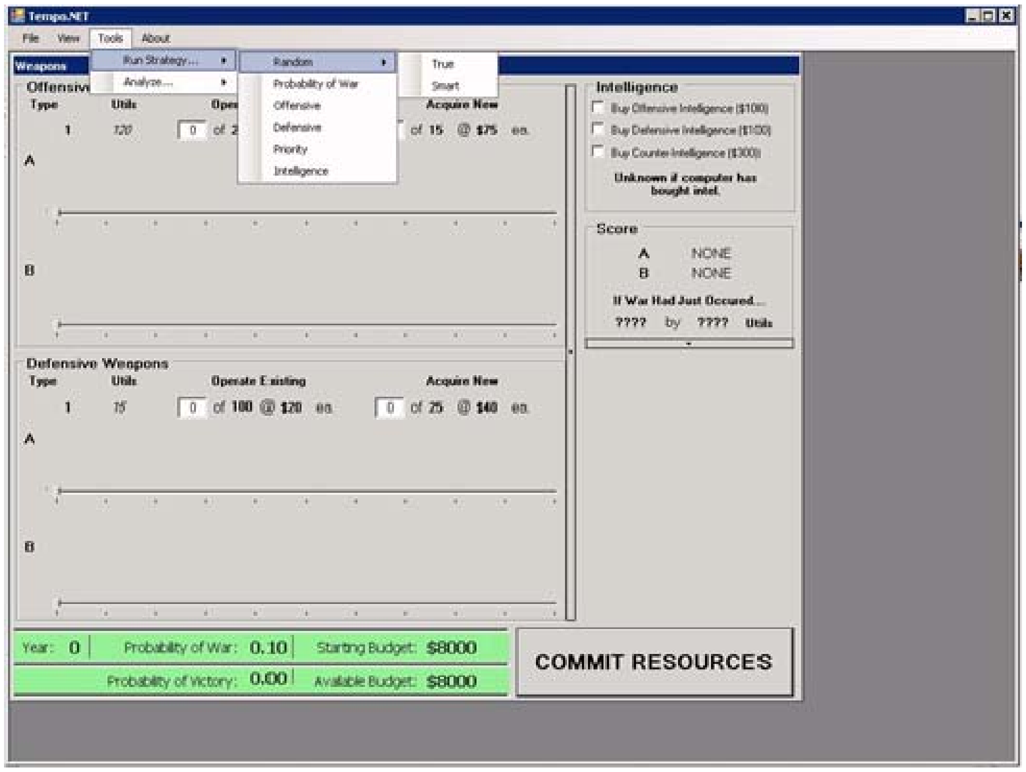
Enter Player Name:

# Trials: 1000

Manual [✓ Automatic]

Pick Strategy: Smart Random

Figure 18. TEMPO Version 3 Strategy Selection Dialog Box



Offensive Weapons

Type	Utils	Operate Existing	Acquire New
1	120	0 of 2	of 15 @ $75 ea.

A

B

Defensive Weapons

Type	Utils	Operate Existing	Acquire New
1	15	0 of 100 @ $20 ea.	0 of 25 @ $140 ea.

A

B

Intelligence

- Buy Offensive Intelligence ($100)
- Buy Defensive Intelligence ($100)
- Buy Counter-Intelligence ($300)

Unknown if computer has bought intel.

Score

A	NONE
B	NONE

If War Had Just Occured...

???? by ???? Utils

Year: 0 Probability of War: 0.10 Starting Budget: $8000

Probability of Victory: 0.00 Available Budget: $8000

Figure 19. TEMPO Version 3 Strategy Menu Options

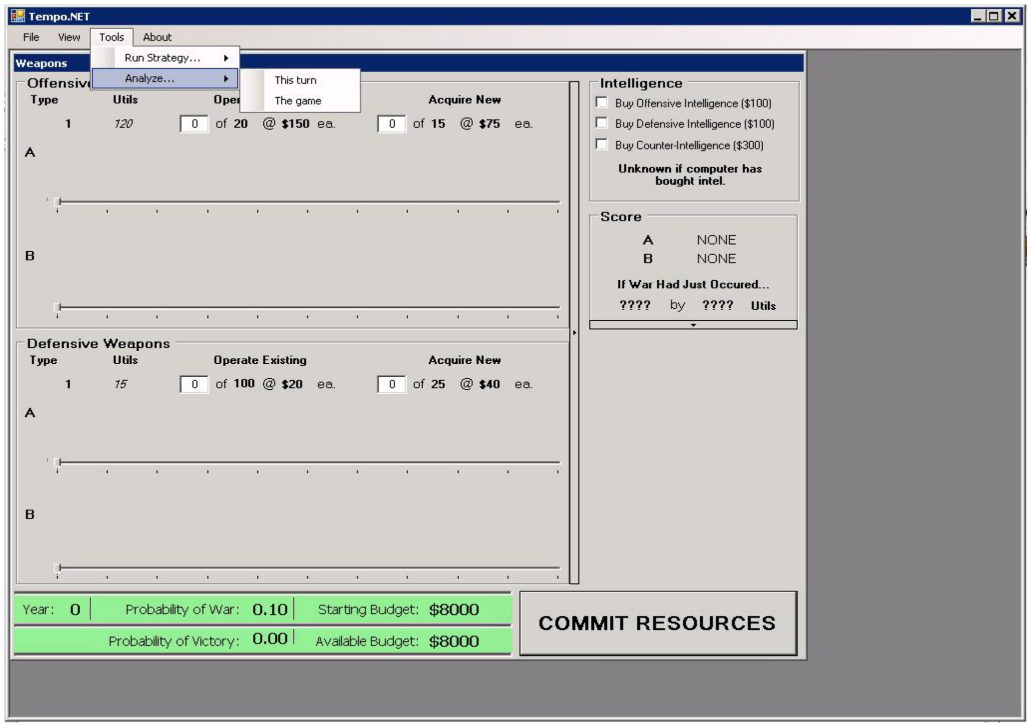


Figure 20. TEMPO Version 3 Analysis Menu Options

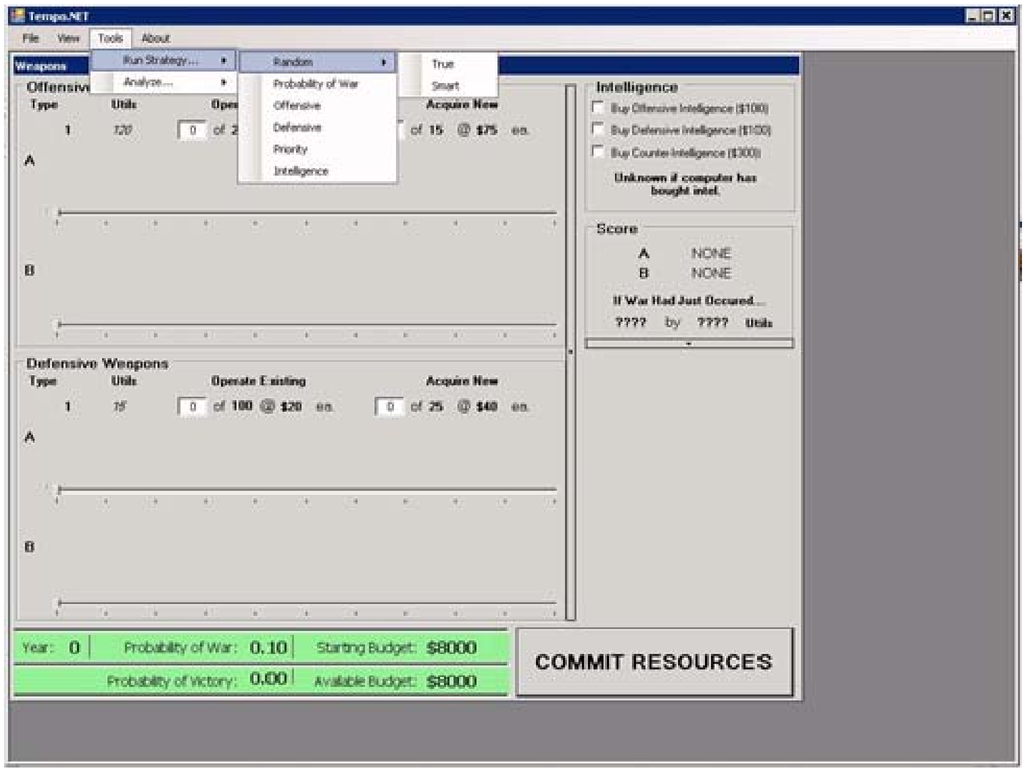
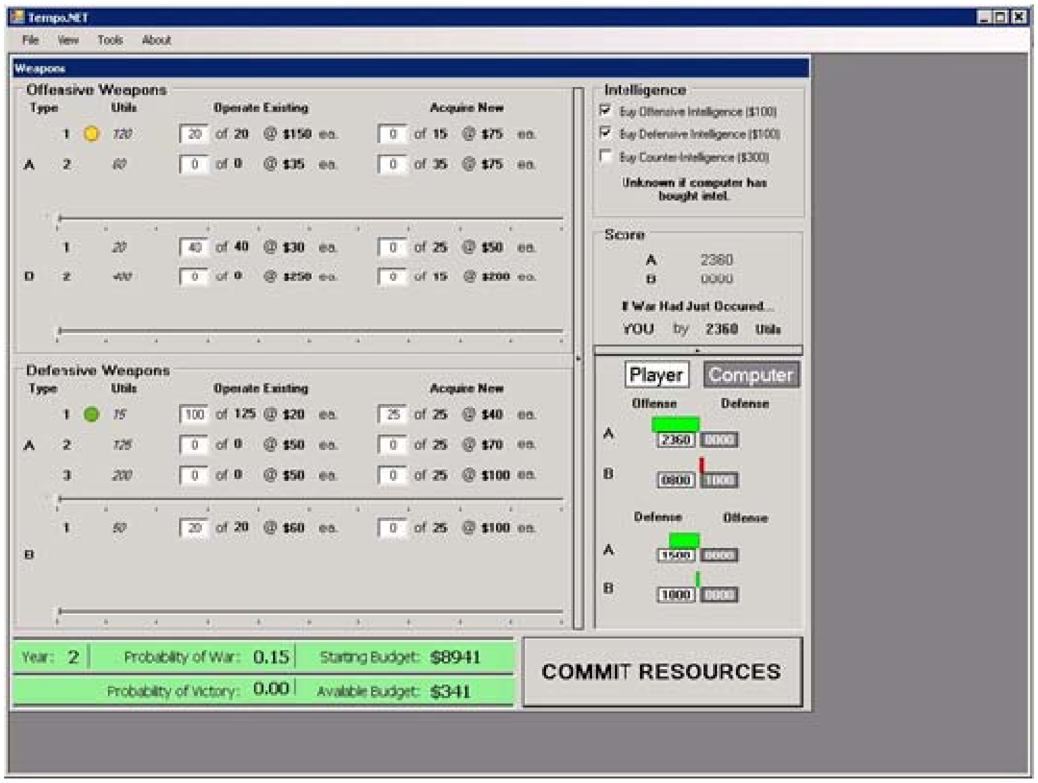


Figure 21. TEMPO Version 3 Calculated Values



Offensive Weapons

Type		Utils	Operate Existing	Acquire New
A	1	120	20 of 20 @ $150 ea.	0 of 15 @ $75 ea.
	2	60	0 of 0 @ $35 ea.	0 of 35 @ $75 ea.
D	1	20	40 of 40 @ $30 ea.	0 of 25 @ $50 ea.
	2	400	0 of 0 @ $250 ea.	0 of 15 @ $200 ea.

Defensive Weapons

Type		Utils	Operate Existing	Acquire New
A	1	75	100 of 125 @ $20 ea.	25 of 25 @ $40 ea.
	2	125	0 of 0 @ $50 ea.	0 of 25 @ $70 ea.
	3	200	0 of 0 @ $50 ea.	0 of 25 @ $100 ea.
B	1	50	20 of 20 @ $60 ea.	0 of 25 @ $100 ea.

Intelligence

[✓] Buy Offensive Intelligence ($100)
[✓] Buy Defensive Intelligence ($100)
[ ] Buy Counter-Intelligence ($300)

Unknown if computer has bought intel.

Score

A 2360
B 0000

If War Had Just Occured...
YOU by 2360 Utils

	Player Offense	Computer Defense
A	2360	0000
B	0800	1000

	Player Defense	Computer Offense
A	1500	0000
B	1000	0000

Year: 2 | Probability of War: 0.15 | Starting Budget: $8941
Probability of Victory: 0.00 | Available Budget: $341

Figure 22. TEMPO Version 3 Score Display

D. MATLAB STRATEGY PACKAGE DEVELOPMENT

Rule-based strategies were necessary to automate the TEMPO execution. These strategies are also necessary to evaluate SEVER. Four main strategy types were derived. They are:

- Strategy 1—Random choice to Acquire/Operate
- Strategy 2—Acquire/Operate based on Pwar
- Strategy 3—Concentrate on one type of force
- Strategy 4—Invest in decisions based on INTEL gathered

Strategy types 1 and 3 contain sub-strategies that the user can further tailor. For example, in strategy type 1, there are two possibilities to choose randomly how many weapons to operate and buy. The first establishes a random number for all weapons available. The second attempts to establish a random number for all weapons and also use the entire available budget for that year. Strategy type 3 can be tailored to focus on operating/buying either offensive or defensive forces. In other cases, it can be tailored to prioritize investing in each particular weapon system based on user preference. Thus, Strategy type 3 contains many strategies within, as each possibility is a permutation of a user-defined priority.

Once SEVER, TEMPO, and all interfaces were well understood, development followed. The first challenge was determining which software packages to use. The current version of TEMPO is a C++ ATL version. In order to input and extract data dynamically (i.e., turn by turn) modifying this code was required. However, the spreadsheet user interface, although a vast improvement over the command console text-based user interface, still lacked simple presentation that a novice user could quickly learn to use. To improve both the user interface, and control of the code, the entire TEMPO model was ported to Visual Basic.NET. All functions were transferred and much testing was performed to ensure the porting successfully captured all critical features available in the C++ ATL version.

In order to execute strategies, MATLAB was selected because of its capability to perform mathematical and numerical manipulation and analysis. It contains many toolboxes to use for analysis and strategy computation. One of interest is the Markov Decision Processes toolbox. Future research could incorporate functions of these toolboxes to improve on the strategies provided within this thesis or generate new strategies to test against TEMPO's coevolutionary code.

1. Strategy Development

The work in this thesis involved updating the existing model to incorporate user-defined strategies, creating those strategies, and providing a capability for the updated TEMPO to accept these strategies. In order to test strategies, it was also required to update the TEMPO model to provide the capability to perform a user-defined number of automatic games (or trials). Data from these trials must be able to be captured and statistically evaluated.

To evaluate performance of each strategy, the strategies must automatically integrate with TEMPO. Six different strategies were coded using the MAThematics LABoratory (MATLAB)^{TM} programming language. A custom MATLAB function was compiled using the MATLAB compiler as a standalone executable file ('runStrategyExe.exe'). During an automatic game, TEMPO calls this function prior to each turn rather than waiting for human input. On the TEMPO side, during initial automatic game setup, the user is able to select the type of strategy and the number of games played ("trials") desired. The interface between TEMPO and MATLAB occurs via three different text files—one to output the human and computer environments from TEMPO, a second to identify to the strategy executable program which strategy was requested by the user, and a third to import into TEMPO the weapon allocation choices executed by the given strategy. These files are named 'tempo.tmp,' 'tempo.ini,' and 'tempo.str,' respectively.

Once all trials have completed, a post-processing tool is used to capture performance data for the strategy. This post-processing application, 'AnalyzeStrategy.exe,' was developed to capture and present the data associated with all the trials run, also using the MATLAB^{TM} programming language. The data is read from the XML log files created by TEMPO, parsed and sorted into weapon allocation variables for human and computer, separately, and output into the MATLAB workspace. Specific plots relating to the performance measures required (see next section) are displayed to visually compare strategies.

A third MATLAB standalone executable file was developed to interact with the user in manual mode on a turn by turn basis. This file, 'graphStrategy.exe' was originally developed to test that each strategy executed by 'runStrategyExe.exe' correctly executed (i.e., applicable budget was used, and proper weapons were operated and purchased based on those available). Through subsequent debugging efforts, the tool can now be used by the player to graphically display the results of the past turn's strategy execution.

The strategies considered to test are described below in algorithmic form. All strategies assume offensive and defensive intelligence is always purchased, however, for this thesis, the only strategy to use this information is the 'Intelligence' strategy. Also, a strategy does not necessarily use the entire budget available each turn. Due to the rules of TEMPO, this unused budget is lost for each subsequent turn. The strategies that do use the entire budget are described as such. The strategies are:

- True Random—Allocation determined randomly
- Smart Random—Allocation determined randomly until total possible budget available is used
- Probability of War—Allocation based on probability of war.
- Priority—Allocation based on concentration of forces of a user-selected order or priority (i.e, Offense A, then Offense B, then Defense A, and lastly Defense B) until the entire budget is used.
- Weapon Type (Offensive or Defensive)—Allocation based on offensive or defensive forces only. Only those weapon types are operated and bought.
- Intelligence—Allocation based on a martingale-style rule to forecast future computer weapon choices.

All strategies are executed through a pre-processing function, 'runStrategyExe.m.' This file accepts two inputs: 1. the 'output file' directory location, and 2. the sub-function directory to call those sub-functions as required. For every strategy, this function:

1. Checks for an existing version of 'tempo.str' in the interface directory and 'tempo.mod' in the temporary write directory. Deletes each, if applicable.
2. Parses 'tempo.tmp' from the interface directory into the appropriate MATLAB workspace variables.
3. Calls the strategy requested by the strategy initialization file, 'tempo.ini.'
4. *--- Executes appropriate strategy as defined below ---*
5. Writes the 'tempo.str' file to a temporary directory within the interface directory from the MATLAB workspace variables 'buyList' and 'opList.'
6. Writes the 'tempo.str' file to the interface directory.

To ensure consistency between the many strategies, a common MATLAB strategy-call function structure was established. The general function structure for all strategies is:

[intelList,opList,buyList] = A_*strategyName*(weaponData,pWar,availBudget,oppData)

In some cases, additional information was required based on the type of strategy. This information is passed first. For example, the 'priority' strategy required the priority of weapons in addition to the other input data. Therefore, its function structure is:

[intelList,opList,buyList] = A_priority(priority,weaponData,pWar,availBudget,oppData)

The data on the right side of the assignment contains all parsed variables passed via 'runStrategyExe.m.' Data on the left side are the outputs of the strategy algorithm, written by 'runStrategyExe.m' into 'tempo.str.' Some variables are not used throughout the strategy execution; however, this common format allows for future customizing with minimal updates required to the 'runStrategyExe.m' functions.

Please note, text below in bold indicates the specific function within the TEMPO-SEVER application-specific functional decomposition satisfied by the particular step in the strategy algorithm. The TEMPO-SEVER application-specific functional decomposition is presented in Appendix B for information.

a. Strategy #1a: True Random

(1) Description. The simplest (to implement) of all potential strategies, the 'True Random' strategy assigns values for the buy and op lists generated using a uniform random number generator. The algorithm creates a 12 x 2 matrix of random numbers between 0 and 1. It then filters 'available' weapons by multiplying the randomly generated 12 x 2 matrix with the binary 'available' vector. Simultaneously, it multiplies the matrix by the inventory and maximum acquisition-able values as reported by TEMPO.NET. These values must range from zero to the number of available 'inventory' and 'maximum acquisition-able' weapons. The end result is a 12 x 2 matrix,

where one column is the ‘opList’ and one is the ‘buyList,’ rounded to the nearest integer ranging from 0 to ‘invent’ or ‘maxAcq,’ respectively.

(2) Algorithm.

1. Generate 12 x 2 matrix of random numbers to serve as ‘opList’ and ‘buyList.’ **(1.3 Develop Information)**
2. Generate ‘opList’ and ‘buyList’
   a. Multiply this matrix by the binary vector of weapons available. This filters out the unavailable weapons.
   b. Multiply this matrix by the inventory and maximum acquisition-able values for all weapons
3. Check to ensure the random allocations for all weapons is within the available budget. If not, regenerate random numbers in same order and execute steps 1 through 3 again. Loop until all allocations are less than the available budget. **(1.3 Develop Information)**
4. Provide results as ‘opList’ and ‘buyList’ to ‘runStrategeExe.exe’ which outputs to the ‘tempo.str’ file. **(1.6 Provide Recommendations)**
5. TEMPO.NET then executes the turn with those weapon allocations. **(2.1 Provide Defense, 2.2 Provide Offense)**

b. Strategy #1b: Smart Random

(1) Description. This algorithm is considered ‘smart’ because it attempts to use as much budget as possible, while still randomly allocating weapon choices. In the case when the budget available is greater than the sum cost of operating and purchasing all available weapons, all units will be operated/ purchased.

NOTE: For the first turn of any strategy (including ‘True Random’) in an automatic game, this strategy is invoked. This ensures the computer cannot falsely win based on a sub-optimal first turn weapon allocation (since all available weapons can be operated and purchased with the available budget). Instead, a tie always results and the next year begins.

The algorithm executes weapon allocation decisions in a pre-determined randomized order calculated by ‘sequenceOrder.m.’ It then randomly assigns the operate/purchase sequence so that for every application of this strategy a different order of weapons is allocated each turn. This ‘randomized order’ ensures a truly random allocation despite the looping algorithm structure.

(2) Algorithm.

1. Gather ‘current environment state’ variable values (weapons Available and associated cost/performance data, probability of war, available budget, and computer utils from last turn if intelligence was purchased). **(1.1 Gather Data)**

2. Determine, randomly, which order weapon systems will be operated (ref. ‘sequenceOrder.m’). **(1.3 Develop Information)**

3. Generate this list of random numbers within ranges of inventory available (for weapon operation) and maximum acquisition-able (for weapon purchase) for all available units, respectively in the order determined from ‘sequenceOrder.m.’

4. Buy/Operate weapons one at a time, in sequence and random numbers previously generated, until the available budget is used (always operating first then buying for each weapon in the order determined by ‘sequenceOrder.m’).

5. Provide results as ‘opList’ and ‘buyList’ to ‘runStrategeExe.exe’ which outputs to the ‘tempo.str’ file. **(1.6 Provide Recommendations)**

6. TEMPO.NET then executes the turn with those weapon allocations. **(2.1 Provide Defense, 2.2 Provide Offense)**

c. Strategy #2: Probability of War

(1) Description. As P_{WAR} increases, weapon allocation shifts from acquire to operate for the highest performing weapons. This value is pre-set as the P_{WAR} threshold. Currently, ‘threshold’ is not settable from the TEMPO.NET interface similar to how the priority and weapon type strategies require additional user input.

(2) Algorithm.

1. Gather ‘current environment state’ variable values (weapons available, associated cost/performance data, probability of war, available budget, and computer utils from last turn if intelligence was purchased). **(1.1 Gather Data)**

2. Determine weapon allocation based on a threshold probability of war occurring. **(1.3 Develop Information)**

3. Establish performance rankings for all available weapons. Operation and acquisition of a single weapon are treated as two separate weapons since they have two separate costs. Performance is based solely on the ratio of Util/$ (ref. ‘rank.m’).

4. Establish probability of war threshold value to switch between operating existing forces and increasing force levels.

5. If the probability of war is less than the threshold, buy or operate units (using ‘activateWeapon.m’) depending on the performance rankings established in 3.

6. Continue operating if probability of war threshold is greater than threshold value, or operate/buy, depending on performance rank, until all available budget is used if probability of war threshold is less than threshold value.

7. Provide results as ‘opList’ and ‘buyList’ to ‘runStrategeExe.exe’ which outputs to the ‘tempo.str’ file. **(1.6 Provide Recommendations)**

8. TEMPO.NET then executes the turn with those weapon allocations. **(2.1 Provide Defense, 2.2 Provide Offense)**

9. Repeat from Step 1 for next year.

d. Strategy #3: Priority

(1) Description. The user establishes a weapon ‘order of preference’: i.e., (OA, then OB, then DA, then DB). Based on available weapons, the strategy always operates, then buys the maximum amount always considering this order of preference until the available budget is depleted.

(2) Algorithm

1. Gather 'current environment state' variable values (weapons available and associated cost/performance data, probability of war, available budget, and computer utils from last turn if intelligence was purchased). **(1.1 Gather Data)**

2. Determine weapon allocation based on user-defined priority of weapons to operate and/or buy. **(1.3 Develop Information)**

3. Convert priority user input strings into binary ID's (i.e., OA = [0,0]; DA = [1,0]; OB = [1,0]; DB = [1,1])

4. For all available weapons, operate first, and then buy the highest priority weapon. Weapon allocation execution occurs by weapon type order (i.e., OA1, OA2, then OA3, if all are currently available).

5. Continue operating, then buying weapons until no more budget available (using 'activateWeapon.m')

6. Provide results as 'opList' and 'buyList' to 'runStrategeExe.exe' which outputs to the 'tempo.str' file. **(1.6 Provide Recommendations)**

7. TEMPO.NET then executes the turn with those weapon allocations. **(2.1 Provide Defense, 2.2 Provide Offense)**

e. Strategy #4: Weapon Type (Offensive or Defensive)

(1) Description. Selection of operate and acquire is focused on one type of offensive weapon system (Offensive or Defensive). This algorithm calculates the best performing weapons (highest Util/$ ratio) and operates them, then purchases, based on the priority given to either Offensive, or Defensive weapons.

(2) Algorithm.

1. Gather 'current environment state' variable values (weapons Available and associated cost/performance data, probability of war, available budget, and computer utils from last turn if intelligence was purchased). **(1.1 Gather Data)**

2. Determine weapon allocation based on user-defined preference for ‘offense’ first, or ‘defense’ first. **(1.3 Develop Information)**

3. Sort all available weapons into Offensive and Defensive weapons.

4. Calculate performance measure (Util/$) for all available weapons using ‘calcEffRatio.m.’

5. Rank operating and purchasing offensive weapons by previously calculated performance measures.

6. Rank operating and purchasing defensive weapons by previously calculated performance measures.

7. Operate, or buy preferred weapon type in order of performance rank. Other weapon preference never gets operated.

8. Provide results as ‘opList’ and ‘buyList’ to ‘runStrategeExe.exe’ which outputs to the ‘tempo.str’ file. **(1.6 Provide Recommendations)**

9. TEMPO.NET then executes the turn with those weapon allocations. **(2.1 Provide Defense, 2.2 Provide Offense)**

f. Strategy #5: Intelligence

(1) Description. Determine weapon allocation based on martingale theory. Paraphrasing Doob (1935), a sequence is called a martingale if each sample, x_n, has an expectation, and if for $m < n$ the expected value of x_n given the past up to time m is x_m. That is, the predicted sample’s expected value is the previous sample’s value (Doob, 1935). Thus, this algorithm bases the expected value of opponent utils for the future turn on the utils reported by intelligence. Then, this rule extrapolates a new util value for the computer based on a linear scaling factor. The scaling factor is called the ‘buffer’ and is required as an input to this algorithm. If the ‘buffer’ is currently coded as 0.10, then the expected value of opponent utils for the future turn are forecasted at 10% above the utils from the last turn.

Based on both the rank of performance measures for each weapon type along with the number of utils forecasted, allocate weapons to cover forecasted utils until all the available budget has been used. (If not all available budget is used and all forecasted utils are operated, continue buying the highest performing offensive weapons. If the entire budget is used prior to completely operating all forecasted utils, the order of operation is DA, DB, OA, OB – the static order is a current limitation of the strategy)

(2) Algorithm.

1. Gather 'current environment state' variable values (weapons Available and associated cost/performance data, probability of war, available budget, and computer utils from last turn if intelligence was purchased). **(1.1 Gather Data)**

2. Allocate weapons based on a forecast of the number of opponent Utils using the 'buffer.' **(1.3 Develop Information)**

3. Calculate the forecasted opponent's utils. (Note: these values for each weapon type are calculated previously to this function call in 'runStrategyExe.exe' and are passed as an input.)

4. Calculate performance measure (Util/$*number available) for all available weapons. This is the utilRatio.

5. Filter the utilRatio, inventory available, and maximum available to purchase into separate variables based on weapon type (i.e., OA, DA, OB, DB).

6. Calculate number of Utils required based on the forecasted opponent's data.

7. Defend up to number of forecasted utils, then operate highest ranking weapons with remaining budget.

8. Provide results as 'opList' and 'buyList' to 'runStrategeExe.exe' which outputs to the 'tempo.str' file. **(1.6 Provide Recommendations)**

9. TEMPO.NET then executes the turn with those weapon allocations. **(2.1 Provide Defense, 2.2 Provide Offense)**

E. MATLAB ANALYSIS PACKAGE DEVELOPMENT

The first step to generate this analysis package was to determine what was required to be verified. At this point in time, verification was only required for the strategy output. In future versions of the package, verification will include the analysis and assessment of SEVER.

Assessment of SEVER would be focused on a set of experiments comparing several automated strategies against the computer. A set number of trials with each strategy would be executed and the outcome captured. SEVER would evaluate each strategy as they execute. The hypothesis could be that the predicted SEVER value calculations for each turn correlate with the statistical performance results of each strategy. Further research could combine the output of SEVER into an additional strategy. This thesis presents the analysis on six general strategies coded in MATLAB. The analysis process can be repeated when the time comes to analyze the SEVER algorithm when executing these six strategies.

But how are the strategies verified to produce consistent, re-producible buy and operate lists? This section focuses on the testing process for each strategy.

1. Strategy Testing

Verification of each strategy involved passing all test case categories. The standard TEMPO.NET output file (‘tempo.tmp’) was manually edited to create that case and the strategy under test was executed. A graph of the strategy’s output for each weapon for each test case was developed to prove verification. Table 1 summarizes the extreme cases tested for all strategies. If the results were consistent with those expected, under all categories, the strategy was verified and a check was placed in the box for that case.

Table 1. Test Case Category Matrix

Weapons available	**Max Budget Available**		**No Budget Available**	
	Enemy Purchased…[2]	**No Intel**	**Enemy Purchased…**[2]	**No Intel**
All	☐	☐	☐	☐
None	☐	☐	☐	☐
OA	☐	☐	☐	☐
OB	☐	☐	☐	☐
Offensive	☐	☐	☐	☐
DA	☐	☐	☐	☐
DB	☐	☐	☐	☐
Defensive	☐	☐	☐	☐

The MATLAB standalone executable ‘graphStrategy.exe’ captures a graphical output of the strategy executed for the latest input file, ‘tempo.tmp.’ This executable was run for each modified ‘tempo.tmp’ file for a total of 16, or 32, test runs depending on the strategy. The output of this function was an allocation graph focused on that particular strategy’s goal. Details on each strategy’s verification are located in Appendix C. A general description of the verification particulars of each strategy are described below.

a. True Random

This strategy is the simplest of all strategies. Verifying this function meant ensuring all available weapons, over a large number of ‘True Random’ executions would generate a uniform distribution of allocation between 0 and the total number of weapons available for operate and purchase for that turn (see Figure 23). The execution of this verification was to run, for each turn, a user-defined number of trials (100) ensuring that any value between 0 and the maximum (for both operating and purchasing all available weapons) was executed. No attention was paid to budget used, except to note that the strategy did not provide an allocation that would result in using more than the available budget.

[2] This case only applies to strategies making use of the Intelligence gathered.

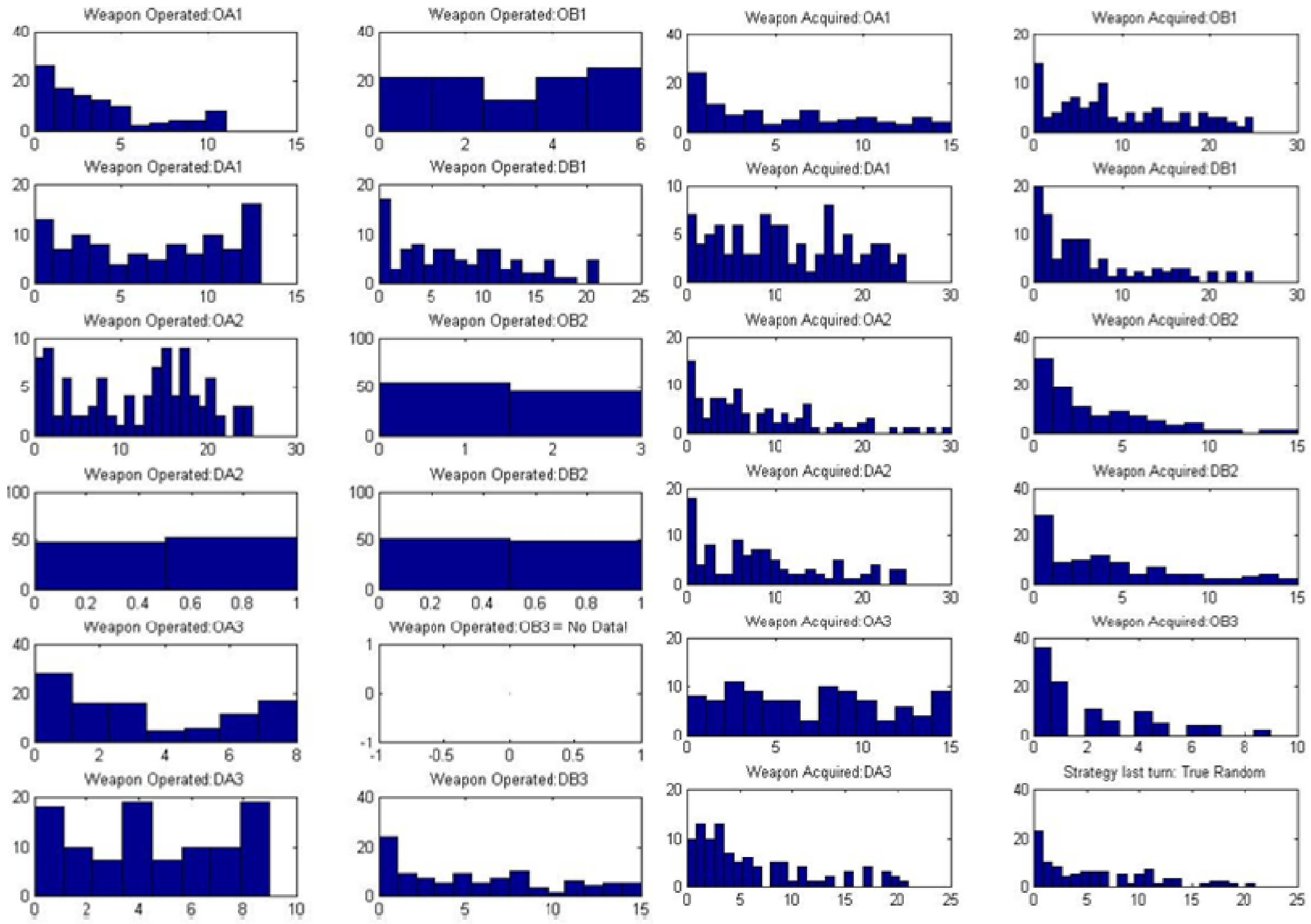


Figure 23. True Random Strategy Verification Graph

b. Smart Random

This strategy built upon the previous strategy by randomly allocating values until all budget was used. The verification for this strategy was focused on ensuring all budget was used until no more available weapons could be operated/purchased (see Figure 24). This was verified by performing a user-defined number of trials (100) for each turn. A bar plot (budget vs. trial) then displayed the results of each trial performed. If any budget remained that could have purchased or operated another weapon, that trial's bar was flagged red. Otherwise, it was green. Thus, the strategy was verified to operate correctly when all bars were green.

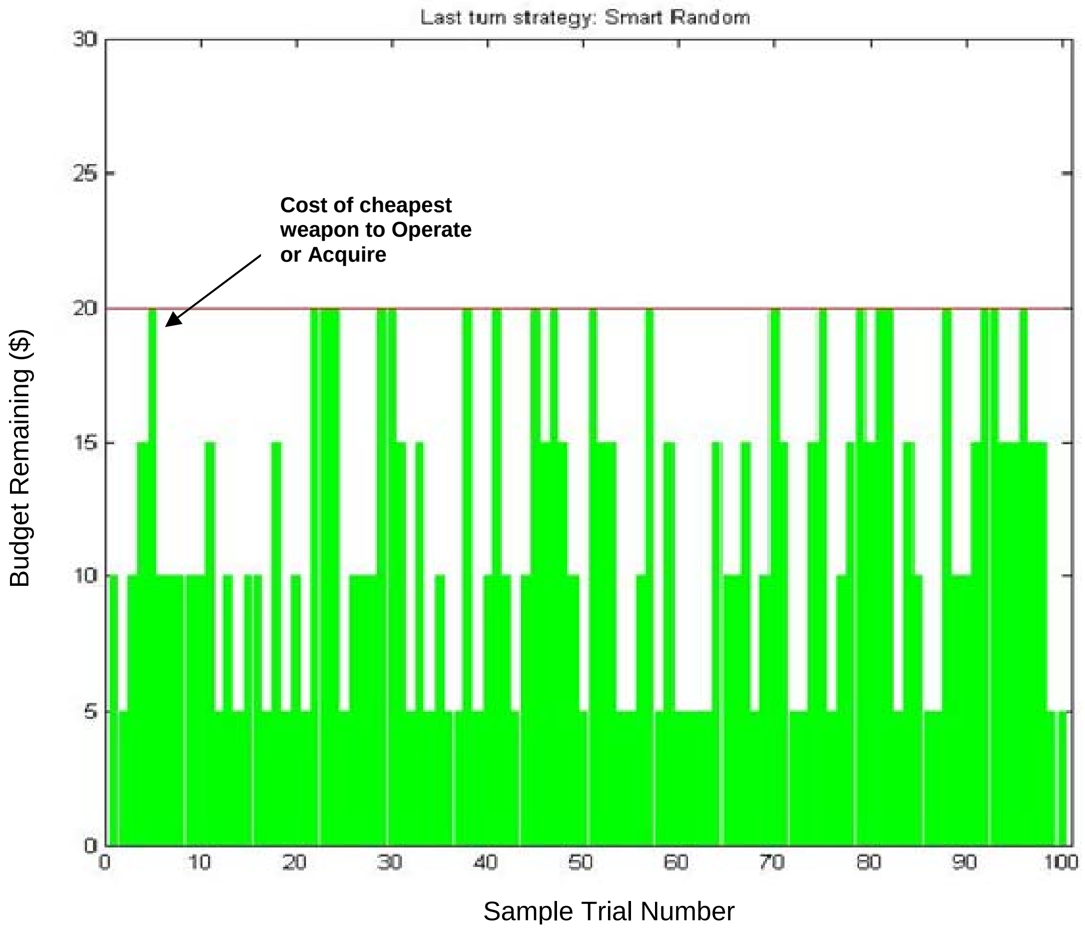


Figure 24. Smart Random Strategy Verification Graph

(1) Probability of War, Priority, Weapon Type. Each of these strategies was verified using the same method. The weapons available and their operate/purchase maximum quantities were recorded. Given these values, for each strategy, the proper allocations for each weapon subtype (i.e., OA1, DB2, etc.) was recorded and coded into the ‘graphStrategy.exe’ function. Next, the actual strategy was executed. Results were stored by the function. Finally, for each strategy separately, the graph displayed predicted and actual on a grouped bar chart side by side (see Figure 25). Many different scenarios were tested by modifying the ‘tempo.tmp’ file to create peculiar situations per Table 1. Once the actual results matched the predicted results for all scenarios attempted the strategy performance was successfully verified.

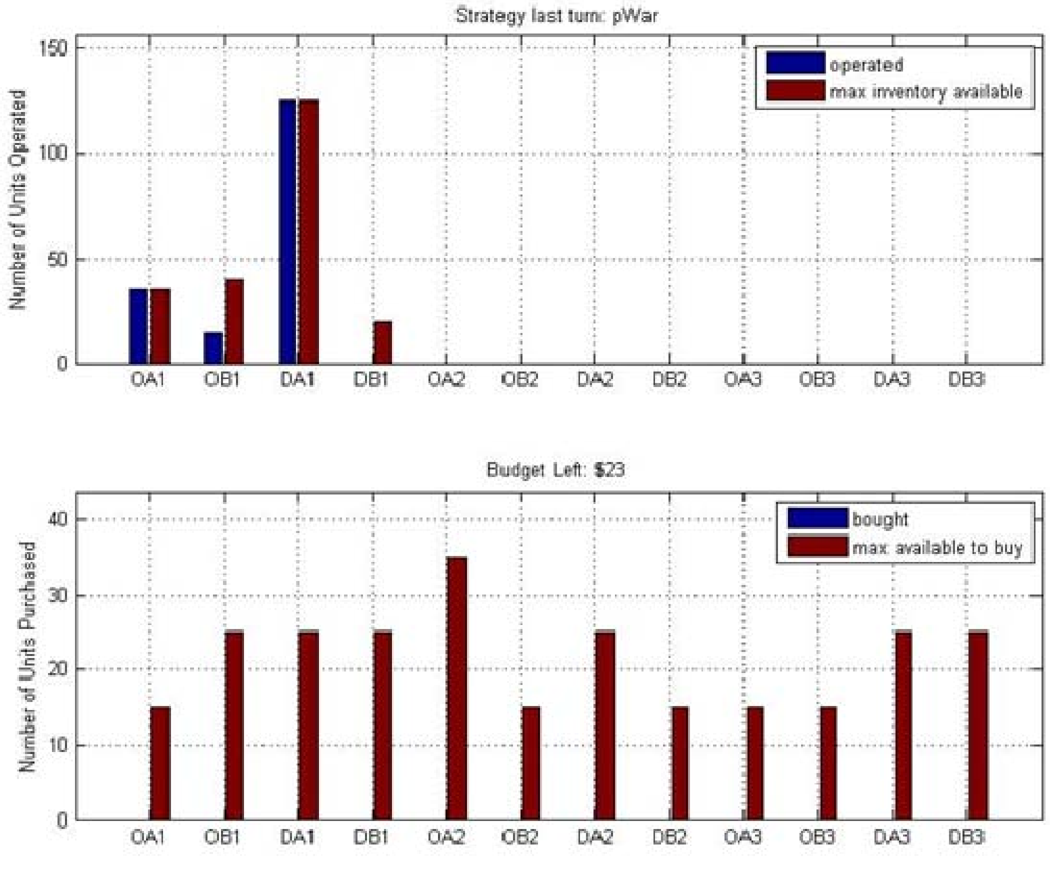


Figure 25. Probability of War, Priority, and Weapon Type Strategy Verification Graph

(2) Intelligence. This strategy was verified using the method described above for Probability of War, Priority, and Weapon Type except for verifying

one additional feature: the use of the opposition's data when purchasing intel. Both the 'Weapons' and 'OppData' sections within 'tempo.tmp' were modified to capture each different case per Table 1. This resulted in more trials performed in order to verify proper strategy execution, but the verification methodology remained consistent. Also, to facilitate verification, the graphical output also displayed the intelligence reported for quick reference between the weapons operated/purchased and the intelligence values captured. See Figure 26 for the intelligence verification graph.

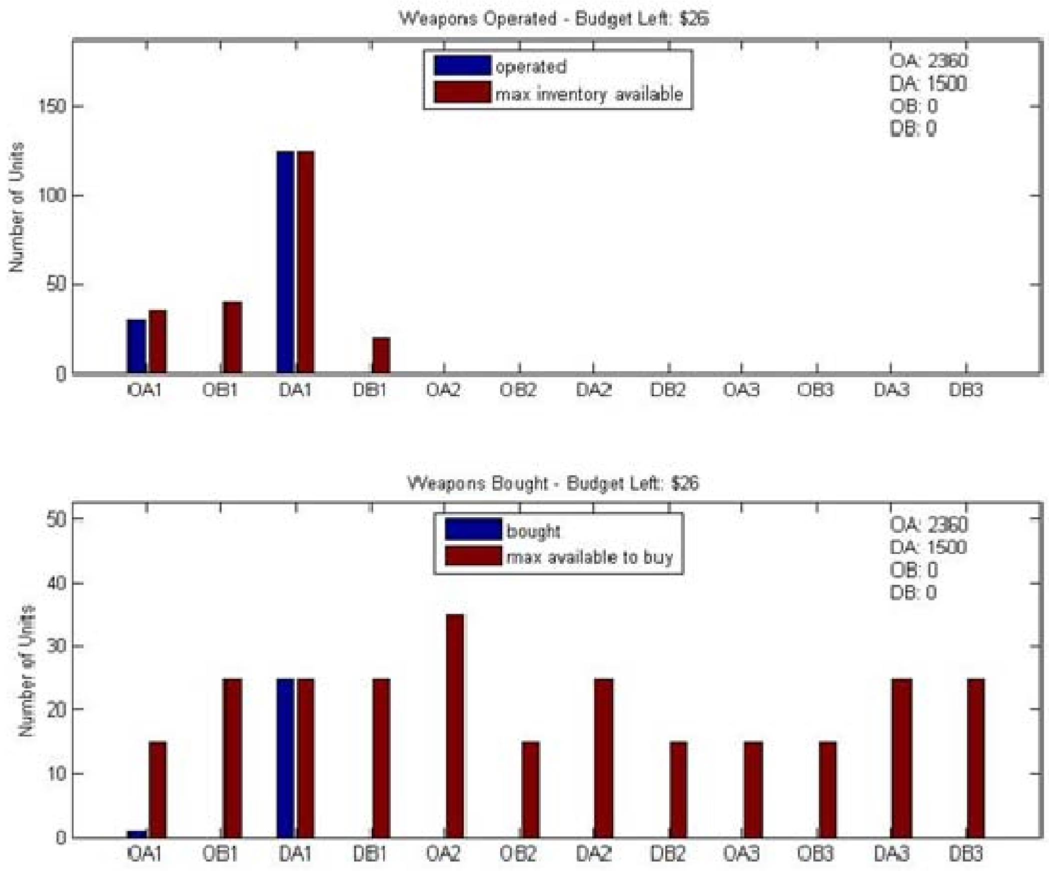


Figure 26. Intelligence Strategy Verification Graph

2. Measures of Performance

These consist of two categories, "per game" and "per turn" measures of performance. Ultimately, SEVER helps evaluate a given strategy's effectiveness at meeting the top-level function "Win game." However, because the game has random variables (i.e., available budget) associated that could potentially provide an advantage to either the player, or computer, a more detailed analysis provides insight into the shortcomings of implementing SEVER.

In this implementation, each strategy was analyzed per each weapon and per total weapons. These measures were also filtered into a per-game outcome and per-turn outcome. This analysis is executed at the lowest level resolution possible by the game. Therefore, for future evaluation using SEVER, it will be possible to answer any potential questions of a particular game or turn. Further research could build upon these performance measures to somehow incorporate these measures as a collective output for use by SEVER.

Per Turn:

- Outcome of each turn (1 for win, 0 for lose)
- Total performance at end of turn (# of util difference)
- Weapon-type performance at end of turn (# util difference)
- Total Efficiency of strategy (# Wasted Utils)
- Weapon-type efficiency of strategy (# Wasted Utils for each weapon type)

Per Game:

- Outcome of each game (1 for win, 0 for lose)
- Performance at end of game (# of util difference)

The performance measures described above are those presented in this thesis. However, because all relevant environment and weapon allocation data for both player and computer is captured turn by turn in the XML performance log files, it is possible to perform post-processing to glean any desired performance measures at time of analysis.

The first research questions state: “Can TEMPO be updated and modified to accommodate SEVER? What is required? How can this be achieved?” In order to evaluate these questions, the development of the TEMPO Version 3 software package commenced. Out of this development effort, more focused questions were derived. These questions include:

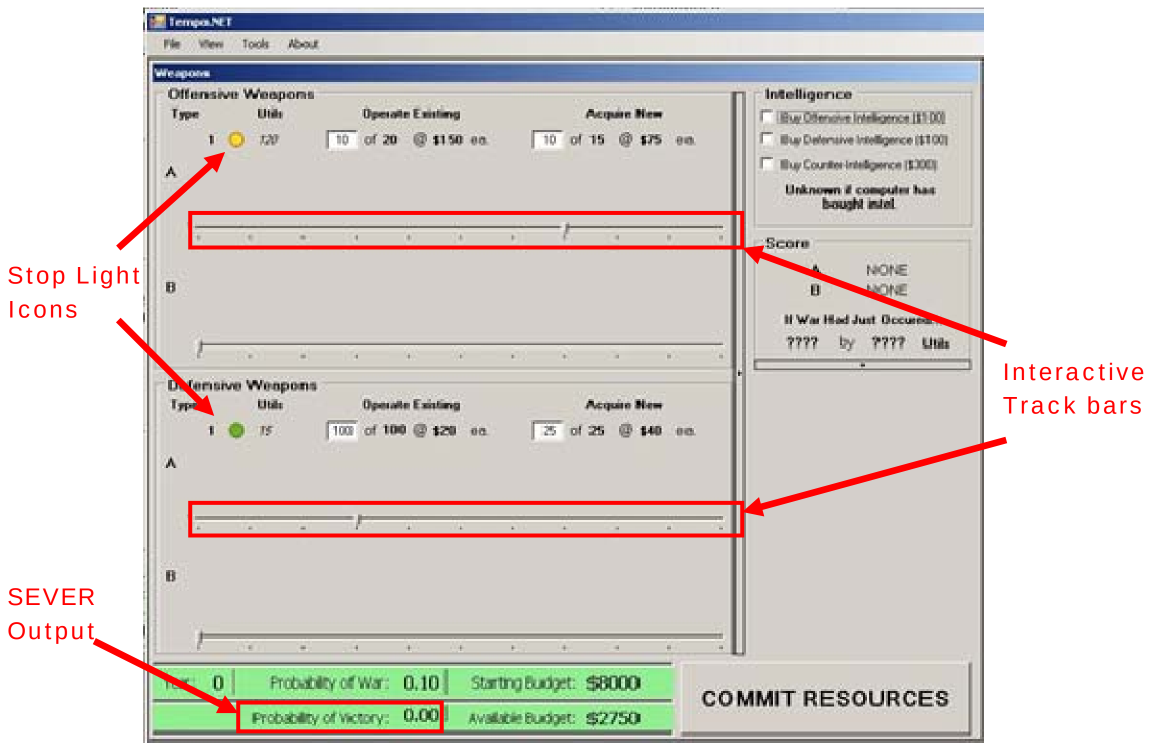
- Are there consistent, predictable patterns for which weapons the computer is likely to operate/buy?
- Is the TEMPO-reported probability of war each turn truly accurate?
- Is the computer smart enough to change its strategy significantly given a certain number of trials against a rule-based player?

The answers to these questions will help determine if TEMPO Version 3, as it stands, is adequate to evaluate SEVER. These questions will be answered in Section G. Results and Discussion

F. INTEGRATING SEVER AND TEMPO

This section describes the general process attempted to apply SEVER to TEMPO. Hypothetical, but realistic cases are presented to establish the envisioned information exchange between TEMPO and SEVER. Work in this thesis does not reflect a complete integration of SEVER and TEMPO since SEVER's notion of 'quality' was not captured. However, TEMPO.NET does include features to facilitate TEMPO-SEVER integration once 'quality' can be properly represented. These are the current SEVER place-holder features:

- Per weapon graphical 'stop-light' icons—Display value as calculated by SEVER as 'high,' 'medium,' or low depending on the user weapon decisions of 'operate' and 'buy' for each that weapon (see Figure 27).
- Per weapon-type interactive track bar—Allows user to select a weight/priority for each weapon type that allows SEVER to dynamically calculate the overall value for the displayed weapon allocations (see Figure 27).
- Per turn total SEVER output—Displays the probability of 'Win game' given 'War Occurs' with the allocations listed at the time. Updates in real-time when allocations are changed to allow the user to affect the probability of success 'on the fly.'



Offensive Weapons

Type	Utils	Operate Existing	Acquire New
1	120	10 of 20 @ $150 ea.	10 of 15 @ $75 ea.

Defensive Weapons

Type	Utils	Operate Existing	Acquire New
1	15	100 of 100 @ $20 ea.	25 of 25 @ $40 ea.

Intelligence

- Buy Offensive Intelligence ($100)
- Buy Defensive Intelligence ($100)
- Buy Counter-Intelligence ($300)

Unknown if computer has bought intel.

Score

A NONE
B NONE

If War Had Just Occurred:
???? by ???? Utils

Year: 0 Probability of War: 0.10 Starting Budget: $8000
Probability of Victory: 0.00 Available Budget: $2750

Figure 27. SEVER Place-Holder Features

G. RESULTS AND DISCUSSION

All the results stated in this section focus on answering the original thesis questions: *Can TEMPO be updated and modified to accommodate SEVER? What is required? How can this be achieved?* As mentioned at the end of Section 0, detailed questions were derived to help answer these questions. The detailed questions are:

1. Are there consistent, predictable patterns for which weapons the computer is likely to operate/buy?
2. Is the TEMPO-reported probability of war each turn truly accurate?
3. Is the computer smart enough to change its strategy significantly given a certain number of trials against a rule-based player?

In order to answer these questions, TEMPO Version 3 commenced so that the applicable data could be captured. This software development effort included the following updates to original code:

- Created an automatic game-play feature that allows a defined number of games to be played with particular strategies
- Developed the rule-based resource allocation strategies
- Updated the GUI to include SEVER outputs
- Conducted many games (trials) to prove the output of pWar is accurate and characterize performance of the computer opponent

The rule-based strategies were verified and compared against one another to demonstrate the statistical evaluating capability of the MATLAB software package. Since all data is captured within MATLAB workspace variables, any desired data analysis can be performed either during, or after all trials and strategies have executed. This allows the user to execute a particular strategy for a defined number of trials and perform performance analysis on those trials without having to see each trial performed. TEMPO Version 3 allows a user-defined data set to be created. For previous TEMPO versions, constructing such a dataset required the user to laboriously perform each trial according to certain rules and extract the data manually into a storage database. This method was extremely time consuming and prone to human error.

After development of TEMPO Version 3 and verification of each rule-based strategy, a data set of 1000 trials was collected for each of 10 sub-strategies. This data set was analyzed to answer the detailed questions stated earlier: *Can TEMPO be updated and modified to accommodate SEVER?*

What is required?

In order to evaluate this question, the TEMPO Version 3 software development effort was executed. Three questions from the TEMPO Version 2 analysis which are important to the updated development:

- Are there consistent, predictable patterns for which weapons the computer is likely to operate/buy?
- Is the TEMPO-reported probability of war each turn truly accurate?
- Is it possible to present performance statistics for each strategy and compare them?

To answer each of these three questions, a determination if TEMPO Version 2 performance and game parameters could be captured and modeled for statistical analysis. By capturing data about each turn of each game including both player's decisions, environmental parameters, and outcome for every turn of every game, the data set would be complete to answer the three questions posed above. Based on the work performed in this thesis, it was determined that TEMPO Version 2 could be updated in such a way to capture all the required data.

How can this be achieved?

TEMPO Version 3 was developed with two premises: creating particular strategy datasets and capturing all decision data, environmental data, and outcome data for every turn of every game for any strategy employed. In order to create this data, an interface between TEMPO and MATLAB was constructed. Automatically, strategies were executed by MATLAB and imported back into TEMPO. Also, the normal (manual) game-play mode was retained with the same data capture and analysis features as for automatic mode. Further, a flexible software architecture to allow for future strategy development was implemented since incorporating SEVER would be performed in a phased approach.

In order to capture the data, the TEMPO Version 2 XML log files were exploited. These log files contain all the required information for every turn of every game. A MATLAB function was created to parse the XML log files output by TEMPO and store the data permanently in MATLAB workspaces. Post-processing is now possible on all existing log files, either game-by-game, or dataset-by-dataset. Also, during manual game-play a feature was added to facilitate the player's decision making by exploiting this MATLAB XML post-processing analysis package and analyzing the interfacing 'tempo.str' file. Regarding SEVER, this feature is very important. SEVER attempts to score a particular decision in real-time. Therefore, TEMPO Version 3 must be able to evaluate data turn-by-turn as well as after a game is over so that this data can be compared against the SEVER-predicted data for both turn-by-turn and game-by-game evaluation of SEVER. Also, this feature could be used for future decision-making research as alluded to earlier.

Understanding what is required to accommodate SEVER into TEMPO Version 2 involves understanding the questions detailed below:

Are there consistent, predictable patterns for which weapons the computer is likely to operate/buy?

In 2005, Johnson, Melich, et al. performed work to adapt the computer strategy by implementing an iterative, coevolutionary learning approach whereby many different decision rules were used by two computer players against each other. Poor performers were eliminated and good performers were retained to give rise to new decision rule sets by "copying," "mutation," and "crossover." Further research could be conducted to implement the same effort with the existing strategies coded into the MATLAB software. Thus, the computer opponent does, indeed, operate and buy with consistent patterns. Characterizing these patterns will help to evaluate the effectiveness of SEVER.

Is the TEMPO-reported probability of war each turn truly accurate?

The true calculation of the "probability of war" value displayed by TEMPO represents the number of games with war in year *n* divided by the number of games reaching year *n*. After all trials were performed, the data were captured to determine after exactly which years war occurs, given the particular year each trial reaches. The resulting calculation is shown in Figure 28. It is evident that there is an anomaly with the 'probability of war' output by TEMPO Version 2, and also TEMPO Version 3 (as this function was directly ported from the Version 2 code). This value is planned to be used by SEVER and therefore must be accurate in order to evaluate the output from SEVER properly.

Is it possible to present performance statistics for each strategy and compare them?

Yes, this is possible since the existing software contains a feature that records the outcome of each game with all the environmental data and allocation decision data for both players. These performance statistics will be used as the baseline statistics against which the predicted SEVER output is evaluated.

As an example, Figure 29 displays the performance of each strategy based on number of *TNO utils*. The red line represents the average value of all data. The blue box represents 95% of the data. SEVER would not be used to predict the actual *TNO utils* as displayed in the chart, but rather the probability of winning the game for a given strategy as demonstrated by the boxes on the plot.

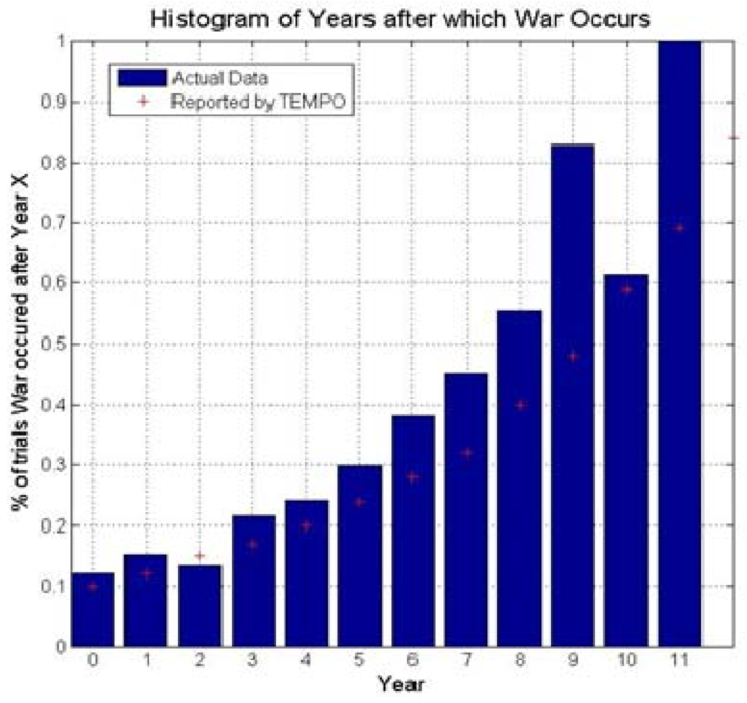


Figure 28. Histogram of 'War Occuring' Year

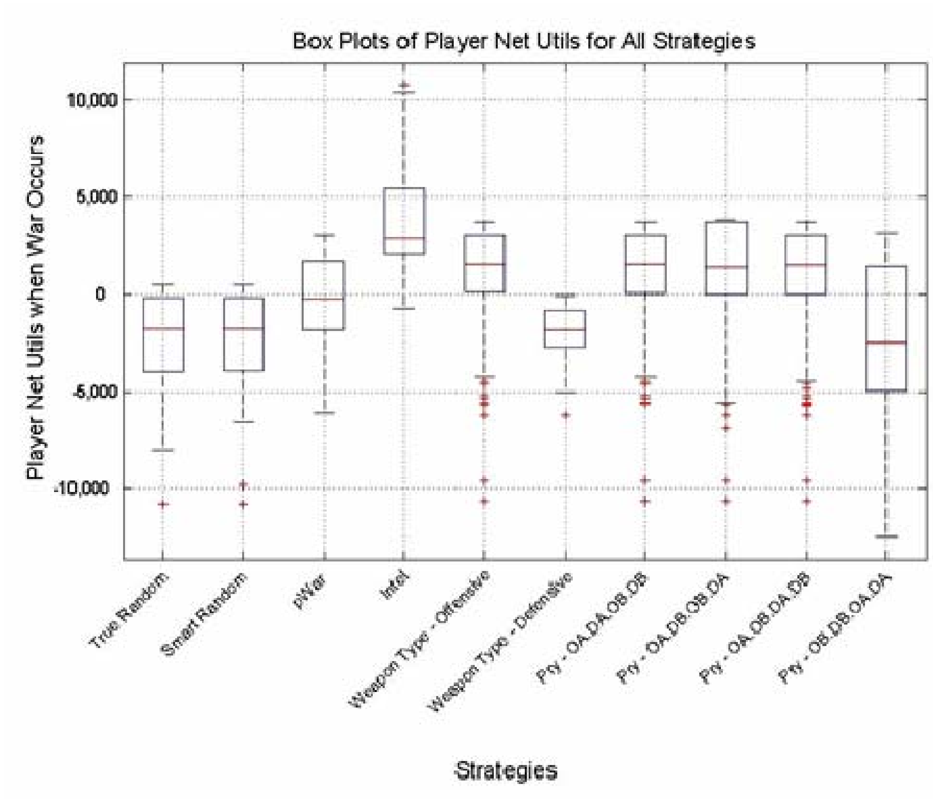


Figure 29. Box Plots of Player *TNO utils* when War Occurs (per Strategy)

Regarding SEVER, one important question still must be answered to properly integrate with TEMPO—how can the SEVER-defined quality be properly represented by the TEMPO variables? The answer to this question is the last item required before TEMPO can be used to evaluate SEVER as a strategy-scoring and prediction algorithm. However, all efforts performed to this point indicate that TEMPO can, and should be used to accommodate SEVER. The TEMPO Version 3 software includes many new features that are conducive for SEVER to function as developed by Langford (2006).

Now that datasets can be captured and stored in MATLAB workspaces, all features of the TEMPO version 2 log files can be used to characterize game parameters statistically. For example, computer decisions for any given turn can be estimated. Also, impacts of a player's strategy can be analyzed for any turn in any game. This allows SEVER's quality variable for a given decision to be successfully modeled.

H. CONCLUSIONS AND RECOMMENDATIONS

Evaluation of SEVER is now possible using TEMPO Version 3. TEMPO Version 3 allows a defined number of trials to be conducted. These results can be plotted in box plot format. There are dedicated areas within the GUI to incorporate SEVER outputs. Any type of strategy can be coded and executed on the TEMPO Version 3 platform. Post-processing of any game variable is available within the GUI. Post-processing of any turn weapon allocation is available, but not yet within the GUI. These new features significantly enhance and automate the type of analysis that can be performed turn-by-turn, game-by-game, and strategy-by-strategy. These enhancements now allow the predicted strategy value determination of SEVER to be compared against actual data.

In summary, the work presented in this thesis confirms the answers to the research questions:

1. Can TEMPO be updated and modified to accommodate SEVER? Yes.

2. What is required? Further questions were derived. In addition to the answers to these questions, it was required to update TEMPO Version 2 to create strategy datasets and capture all data.

    a. Are there consistent, predictable patterns for which weapons the computer is likely to operate/buy? Yes, there are.

    b. *Is the TEMPO-reported probability of war each turn truly accurate?* At this time, it has not been confirmed that these values are truly accurate. Evaluation of this aspect of the TEMPO Version 3 VB.NET code is required for future integration and evaluation of SEVER.

    c. Is it possible to present performance statistics for each strategy and compare them? Yes. This is demonstrated in Figure 29.

3. *How can this be achieved?* Further questions were derived. Answers to these questions summarize how it is possible to accommodate SEVER into TEMPO Version 2.

    a. *How should TEMPO Version 2 be modified?* Create a capability to run strategies automatically and store all environmental data, player and opponent data, and per-game outcome data. Design in the flexibility to add strategies to the software to further characterize the computer's performance so that the SEVER representation of quality can be incorporated. Currently, this feature is only implemented via MATLAB text-based functions run on a particular MS Windows Explorer folder containing the XML log files. There is a placeholder for this feature, as shown in Figure 20. (ref. 'The Game'). Also, allow turn-by-turn analysis to be executed since SEVER can be used to evaluate a single turn's weapon allocation strategy along with a particular game's overall allocation strategy. This feature is also implemented via MATLAB, but can be activated through the TEMPO Version 3 interface as shown in Figure 20 (ref. 'This Turn')

b. *Can the TEMPO game be modified to evaluate game play resource allocation decisions while the game is played (i.e., in 'real-time')?* Yes, this is possible with the use of 'tempo.str' as the data for analysis rather than the XML game log file. As previously mentioned, this feature can be found in the software as shown in Figure 20.

Other than the required update to incorporate SEVER: characterize the TEMPO performance to evaluate quality, the author suggests some software improvements. The following software improvements are suggested to increase the time it takes the in-game strategy and analysis functions to execute:

- Replace MATLAB standalone executable called from TEMPO.NET with a MATLAB DLL containing all associated functions and code directly into TEMPO.NET. This would eliminate the continuous search loop required to scan for 'input files.'

- Create the GUI for the MATLAB analysis code for the XML Log file to streamline the post-processing and analysis of game log files.

- Create a GUI for the MATLAB analysis code independent of the TEMPO GUI so that post-processing can be executed simply.

- Perform true statistical analysis using the MATLAB Statistics Toolbox on the existing, and future, strategies.

These suggestions are intended to improve upon the features and capabilities that already exist in TEMPO Version 3. The new features of TEMPO Version 3 are the first step in improving the human's decision process. These features are an attempt to consider a high-level management resource allocation decision (in this case, budget) and provide a total systems approach to collect, organize, and summarize the impacts in an automated fashion using today’s powerful computational processing. This saves the user time because he/she does not have to physically and mentally organize this information himself/herself. Therefore, more effort can be spent on assessing a particular outcome’s consequence. Based on the user’s preference, select information can be accessed in real-time to allow quicker understanding of a given decision’s impacts just after it is executed. If successful, the incorporation of SEVER will further build on this capability by being able to predict the outcome prior to executing the decision.

Ultimately, the goal is to automate a DoD resource allocation decision process based on SEVER-predicted outcomes and applicable presentation of the right information at the right time. Whether the decision is a “satisfaction” decision involving, strategically, how forecasted budget should be allocated toward future versus current military capabilities, or an “optimization” decision involving what the optimal tactical force structure is required in a certain forecasted conflict, the software tool presented in this research, along with the SEVER algorithm, can one day be adapted to present the required information at the right time to the eventual decision maker.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX A. VALUE SYSTEMS ENGINEERING (GARY LANGFORD)

Appendix A was prepared and written by Professor Gary Langford, my advisor at the Naval Postgraduate School. It summarizes the metrics of Value Systems Engineering, in so far as it relates to a framework that can be used to further the evaluation of SEVER as a game theoretic. The included references cite works that are integral to Value Systems Engineering which form the foundation of this thesis.

This section formulates and applies a simple rootage for Value Systems Engineering (VSE). This conceptualization of VSE embodies the same general notions of systems, system elements, functions, performance, and quality, but redacts many of the fundamental approaches commonly used (Langford & Huynh, 2007) in Systems Engineering. For example, the essence of a *system* is a set of elements that are either dependent or independent but interacting pairwise—temporally or physically—to achieve a purpose. Elements that only interact directly with other system elements are internal to the system. Elements that interact with both internal and external elements form the boundary of the system. We include the permanent and episodic interactions among elements of a system, systems of systems, and a family of systems. A system thus includes the lasting and occasional interactions, as well as emergent properties and behaviors. The interactions between elements effect transfer of energy, e.g., materiel, data, information, and services. The interactions can be cooperative or competitive in nature, and they can enhance or degrade the system value, which is defined below.

A. FUNCTION

We define the worth of a system (or product or service) in terms of the system functions, their performances, and their qualities (i.e., a Taguchi loss function). For example, a product shall provide a function with specified performance and a delimited level of quality. A function is an action performed by the system that is required to achieve a system objective. System functions may change and be added or deleted. The concepts of performance and quality of a function will be elaborated in the following discussion.

B. VALUE

Value (V) per function is defined as the ratio of performance (P) to investment (I), the fundamental premise of Value Engineering (Miles, 1972). Value compares what one receives with what one has invested. If there are two products with factually comparable features offered for different prices, the value of the lower-priced product is higher than that of the other product (Langford, 2006). The value of a system is measured by its worth (the actual and expected use of a product or service) relative to the investment made in obtaining the system. The system value may vary with time. To account for additional investments made during the system lifecycle, the investment can also change with time. The Value Engineering equation, which relates performance and cost, can be rewritten to include their implied relationship to the same function(s).

The system value, V(t), is given by

$$V(t)_{\Sigma F(t)} = \sum \frac{P(t)}{I(t)} \tag{A.1}$$

where $F(t)$ is a function or non-linear summation of functions that are performed by the system, $P(t)$ is the performance measure (units of energy) of the function(s) $F(t)$, $I(t)$ is the investment (e.g., dollars or other equivalent convenience of assets that are ‘at-risk’) and the time, *t,* measured relative to the onset of initial investment in the project. The investment can be incremental and summed to equal the lifecycle investment or partitioned and equal to a unit or item investment. The units of *V(t)* can be expressed in terms of energy divided by cost. We refer to the delineation of a function in terms of its performance, and the quality of that performance, as the triadic decomposition of the function $F(t)$. The summation in Equation (A.1) is simplified for the purposes of this discussion, and thus shown over all functions, performances, and investments.

This construct of value is used by Value Engineers to answer the question, “Why does something increase the cost so much”? Value analysis is a problem-solving system that assists in the application of better approaches, alternative materials, appropriate processes, and in identifying the advantages of various suppliers. The point of Value Engineering is to be more effective in reducing costs without compromising on satisfying the customer.

C. PERFORMANCE

Performance indicates how well a function is performed by the system. Performance is an objective measure of its related function. In this general construct and application for software piracy, quality refers to the consistency of performance (or designated tolerance that signifies the deviation allocated to the performance requirement) in reference to the amount of pain or loss that results from the deviation as described by Taguchi (2005).

The change in performance of a system element due to the transfer of energy from another element is equal to the work done. Performance is accomplished with reference to the cost/unit time as well as to the total time over which the performance occurs. Incorporating the variable of time and then factoring it results in expressing the value equation in terms of the metric of *performance* per rate of investment (e.g., spending).

$$V(t)_{\Sigma F(t)} = \sum \frac{P(t)}{I(t)/t} * \frac{1}{t} \tag{A.2}$$

In essence, this formulation of Value Systems Engineering implies that functions result in capabilities; performances differentiate competing products; and quality affects the lifecycle cost of the product. For each function, there is at least one pair of requirements—a set of performance requirements for each function and a set of quality requirements for each performance requirement.

D. QUALITY

The quality requirement indicates the variation and impact of that variation on the performance requirement of a function. Quality indicates how well a function is accomplished (through its performance) by the system. It is a measure of the variation and impacts of the variation of the performance requirement(s), or that performance achieved by the system, associated with its related function. Additionally, quality is a measure of the loss due to the performance of the system. The performance requirement is measurable and testable. The quality requirement derives from the view of the system throughout its lifecycle and characterizes the system losses due to predetermined functions, i.e., non-delivery of the system's functionality, or operations beyond the range of specified performance tolerances. A system function may thus have any number of performance parameters and, likewise, several quality requirements associated with a given level of performance.

E. WORTH

Worth (*W*) is the use of a product or service as represented by the functions and their related functional attributes—performance, quality, and investment. Multiplying Value from Equation (A.2) by Quality is defined as the Worth of $F(t)$. Additionally, multiplying the numerator and denominator by *P(t)* defines the performance metrics and indicates that quality is a measure relative to performance.

$$W(t) = [V(t)_{\Sigma F(t)} * Q(t)] = \sum \frac{P(t)}{I(t)/t} * \frac{P(t)}{t}\frac{Q(t)}{P(t)}] \qquad \text{(A.3)}$$

where $Q(t)$ is the quality (which can be considered as a tolerance assigned to $P(t)$). Stipulating the units of $Q(t)$ to be the same as that of $I(t)$, determines the unit of *W(t)* to be that of $P(t)$, since $F(t)$ is dimensionless. The summation in (A.3) is simplified for the purposes of this discussion, and thus shown over all functions, performances, quality, investments and temporal notions. Equation (A.3) times likelihood is referred to as the Systems Engineering Value Equation with Risk (SEVER). Risk is discussed in section g.

The interaction between elements transfers a measure of worth (defined as value convolved with loss) from one element to the other element of the pair of elements. We term the measure of the transferred worth the Worth Activation Function (*WAF*).

F. WORTH ACTIVATION FUNCTION

The *WAF* is a basis for analyzing a system. In control theory, a transfer function is a mathematical representation of the relation between the input and output of a system. A *WAF* between two elements of a system is defined to be the exchange of value between the two elements. The elements exchange energy and one measures performance. Value is that performance achieved for a given investment. This exchange necessarily assumes some measure of risk. Given risk, a *WAF* can thus be either a manifestation of the state, (or a change in state of a system) or a tool to evaluate differences between the state of a system and the state of another system or between the states of two systems in a system of systems. In essence, the *WAF* represents various impact(s) on the state(s) of a system. The *WAF* can be a nested hierarchy of *WAF*s, all related through the triadic decomposition of functions, performance, and quality. Depending on the value ascribed to each of the *WAF*s, the state(s) of the system(s) may be impacted to varying degrees. The result is that a small number of *WAF*s may be equivalent to a large number of irreducible *WAF*s. A small number of highly decomposable *WAF*s may be equivalent to a large number of irreducible Worth Activation functions.

G. RISK

Using the logic in from Lowrance (1976), Lewis (2006) defines simple risk as a function of three variables: threat, vulnerability, and damage. Replacing damage with worth, Langford and Horng (2007) capture risk through threat, vulnerability, and worth. An element e of a system is associated with a risk, R_e defined by

$$R_e = X_e U_e W_e = X_e(1-a_e)WAF_e \tag{A.4}$$

where, threat, X_e, is a set of harmful events that could impact the element; vulnerability, U_e is the probability that element e is degraded or fails in some specific way, if attacked;

worth activation function, WAF_e, results from a successful attack on element e; and susceptibility, a_e, is the likelihood that an asset will survive an attack. WAF_e is given by Equation (A.3). It may be loss of productivity, casualties, loss of capital equipment, loss of time, or loss of dollars. Susceptibility is the complement of vulnerability. Susceptibility is loosely defined as the inability of a strategy to avoid being countered in a game-competitive environment, whereas vulnerability is the inability of the strategy to withstand countering caused by the threat. Susceptibility and vulnerability can be measured by the probabilities of these events happening. Therefore, the probability of strategy surviving a game-competitive environment (strategy survivability) = 1 – Probability of the strategy being countered (susceptibility) x Probability of the system succumbing to the effects of the strategy (vulnerability).

Since an element in a system (or network) may be connected to more than one element, the number of *WAF*s associated with the element is the degree of the element. Subscribing to Mannai and Lewis (2007), we obtain the system risk, R, as

$$R = \sum_{i=1}^{n+m} X_i (1 - a_i) g_i WAF_i \tag{A.5}$$

in which n denotes the number of elements, m the number of links or *WAF*s, and g_i denotes the degree of the i^{th} element. As a result of the *WAF* between two elements, e_1 and e_2, at the moment of their interaction for some elements, we have

$$\frac{WAF_{e_1}}{R_{e_1}} = \frac{WAF_{e_2}}{R_{e_2}} \tag{A.6}$$

It is this expression in Equation (A.6) which forms the basis for understanding transactions between elements (i.e., exchange of energy) that are independent and arms-length (e.g., buy-sell arrangement between unrelated parties) within a particular system for which the Worth Activation Function is defined (Langford et al. 2007). Equation (A.6) is also the basis for defining and evaluating complexity as well as interpreting and predicting emergent properties of systems. Complexity and emergence are determined by numbers of elements, their Worth Activations Functions, and probability that the Worth Activation Functions will be at the Pareto-optimum value(s).

H. DECISIONS BASED/CAPTURED BY SEVER

Decisions (from the perspective of the human game player) are represented in Equation (A.4) as the set of potential losses, i.e., a reduction in the quality of a follow-on decision for a given level of the performance achieved as a consequence of a former decision. For the high quality decisions there are also other losses that accrue, such as game-play length of time for the turn. From Equation (A.4), the losses from a decision can be represented as shown in Equation (A.7).

$$Disruption\,Due\,To\,Low\,Quality\,Decision = \frac{Q(t)}{P(t)} \tag{A.7}$$

These losses are typified by a loss function of a quadratic form for game-competitive environments. From Equation (A.7), the highest quality decision from the perspective of the human game player's value chain is represented without the disruptive factor, Equation (A.8) as:

$$High\,Quality\,Decision = [V(t)^{\Sigma F(t)*} Q(t)] = \sum \frac{P(t)}{I(t)/t} * \frac{P(t)}{t} \frac{Q(t)}{P(t)}] \tag{A.8}$$

I. RAPID SYSTEMS ENGINEERING

We apply the general methodology of Rapid Systems Engineering (Langford, 2006) to explore the Worth Activation Function's general utility to characterize software piracy through its Stakeholder Analysis Methodology. Rapid System Engineering (RSE) is a scenario-driven approach that attempts to reduce the degree of uncertainty in predicting enterprise success by structuring and analyzing the interplay between alternative operational models, competitive strategies, and their resultant product alternatives. The RSE structure is a bottom-up, systematic, and highly iterative set of steps that marry competitive strategies to alternative operation's requirements and conditions for stakeholder success. Applying RSE to software piracy presupposes that the four attributes of a software pirate's successful business operation are satisfied. These attributes are first, the business value proposition articulated (the reason customers buy the pirate's products and compensate the pirate. This could imply the software pirate is in the business of making a profit of the labors of others or that the software pirate is satisfied through other means or vicarious extensions. Second, the software pirates have identified market segment(s) or groupings of customers who are aggregated in a common distribution channel or some other segmentation that results in an economy of distribution that is acceptable by the pirate. Third, the structure, activities, and processes that comprise the pirate's practices that contribute to the worth are defined. These include the value chain and its logical relationships of low-order separation. Fourth, the mechanics and venues of generating revenue are described and tractable.

APPENDIX B. AN EXAMPLE APPLICATION OF SEVER

A company wants to provide an efficient lighting system for office buildings. From a systems perspective, "providing and efficient lighting system" consists of more than lighting a building. At a minimum, it consists of:

planning efforts—physical work required to properly design, test, integrate, and support the system to meet customer expectations

development efforts—physical work to research concepts, communicate ideas, procure hardware, validate design, and integrate the system

installation efforts—physical work required to package the system, ship the system, install the system, and test the system

operation effort—daily actions and costs by the consumer to operate the system

maintenance and support efforts—costs and time associated with maintaining the system, preventing system failures, responding to system failures, and ensuring the system meets the customer's daily expectations

disposal efforts—removal costs, environmental impacts, time to dispose, resources required to dispose

Can this company set themselves up for success? The answer to this question can be approached by applying SEVER. First a functional decomposition of this business is performed. To simplify the example, one particular function in the "development efforts" phase, F_1, "Illuminate workspace" is considered. For completeness, the reader should note that the sum of each function's SEVER-calculated value is the total system value. Thus, the following process can be carried out for each function in the functional decomposition, and a total system value can be calculated. The value of the first function, $F_{1,}$ is stated, algebraically, as:

$$V_{1F1} = \frac{P}{\$/t} \bullet \frac{Q}{P} \bullet \frac{P}{t} \tag{B.1}$$

The three grouped terms, $\frac{P}{\$/t}$, $\frac{Q}{P}$, $\frac{P}{t}$ reflect realistic quantifications of performance, quality, and investment. *P*, *Q*, and *I* must be determined to properly calculate the value for function, $F_{1.}$ If *P* is considered 'lumens', *$/t* can be considered '$ paid/hour of operation'. *Q/P* can then be considered '$ lost due to poor quality [during operation]/lumen'. Lastly, *P/t* can be defined as 'Total # lumens operated/total hours of operation'. The most difficult term to quantify is the quality term. This term is based on Taguchi's quality—a loss to society resulting from less-than-optimal quality. In this case, the poor quality can be represented by: lighting a building without occupants, non-optimal brightness, flickering lights, and unreliable illumination, to name a few. The effects on society from these situations are many. A few short-term loss examples include less than optimal working conditions, disgruntled employees, and wasted electricity. A few long term examples include an unmotivated workforce and employees requiring vision insurance. The most important thing to note in this example is the effect of poor quality, as defined by Taguchi, at some point, affects society as a whole (Taguchi, 1990). Certainly, a preliminary analysis not considering quality would not have considered a vision insurance company related to the performance of a lighting system.

Now, the equation reads:

$$V_{1F1} = \left(\frac{lumens}{\$\,operated / hour\ of\ operation} \right) \bullet \left(\frac{\$\,lost\ due\ to\ poor\ quality}{lumen} \right) \bullet \left(\frac{Total\ \#\,lumens\ operated}{total\ hrs\ of\ operation} \right) \tag{B.2}$$

The value of this function is represented in units of *lumens*. This example, although not a complete technical breakdown of SEVER, illustrates an important feature of SEVER—the ability to plan and model, from a bottom-up approach, while being able to ultimately execute the business, or strategy, in a top-down manner.

APPENDIX C. SERVER APPLIED TO TEMPO FUNCTIONAL DECOMPOSITION

The first level of the functional decomposition (after much iteration) is shown in Figure 30. The “home plate” symbol represents further decomposition not reflected in the specific figure. Figures 31–34 represent the further decomposition of function *1.0 Analyze Situation*. Figure 35 represents the further decomposition of function *2.0 Play Game*.

Questions that were considered during functional decomposition:

- During gameplay, is it better to consider operating total utils independent of computer’s number of utils?
- Should the user break down their decision making to concentrate on one weapon type at a time?
- Should the user consider ‘balancing’ the portfolio of weapon choices or operate one type in particular?
- Is there a difference between offensive and defensive Utils?
- Does purchasing Intel provide an advantage?
- Does purchasing counter-intel provide an advantage?
- Each weapon type differs in performance (i.e., some offer more utils than others). Each weapon type also differs in investment required (i.e., some are cheaper than others). Therefore, the value of each is weapon system is different. Is value dependent on total amount available to spend?
- Is the goal to distinguish between competing weapons of the same type (i.e., OA1 vs. OA2 vs. OA3) or competing weapon types themselves (OA vs. OB, etc.) or both?

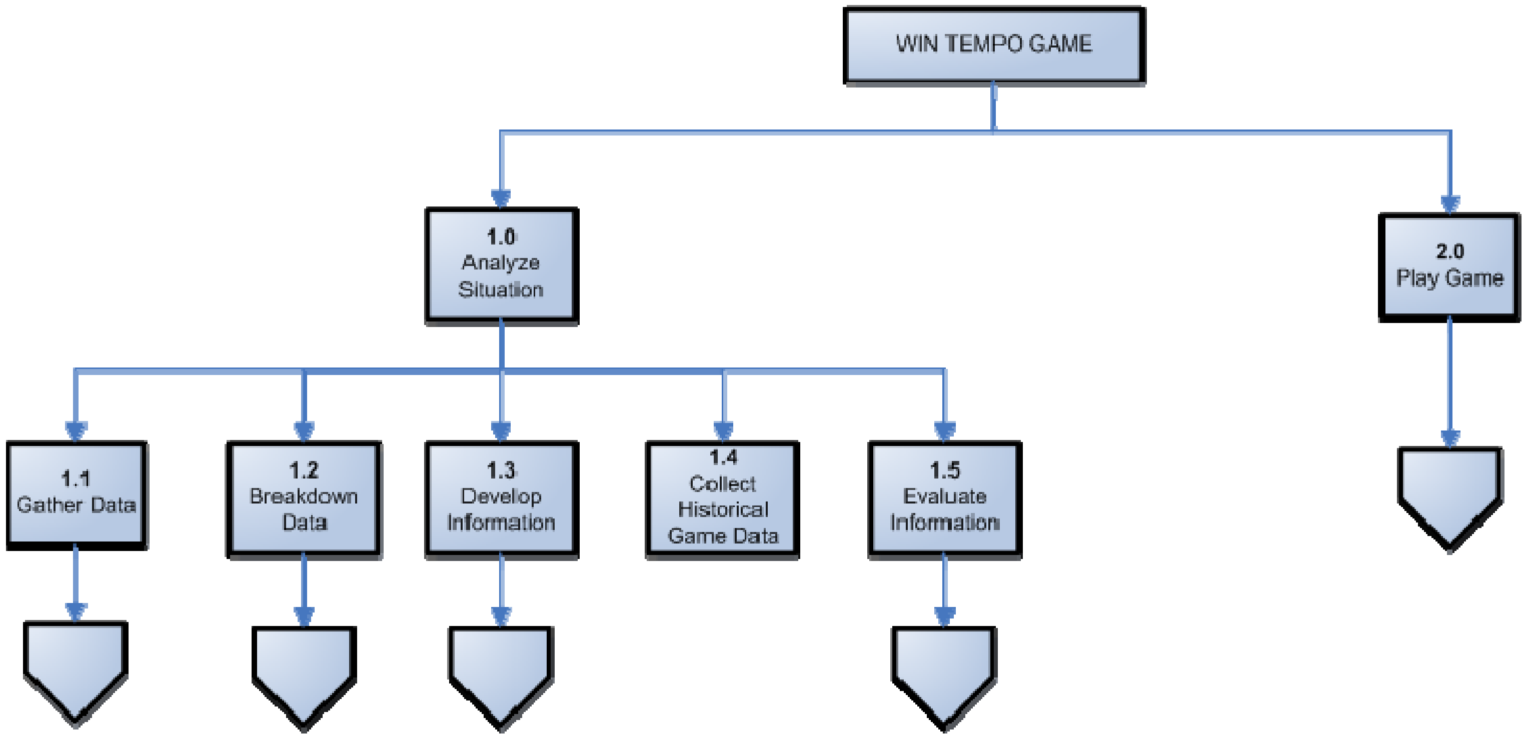


Figure 30. Top Level Functional Decomposition

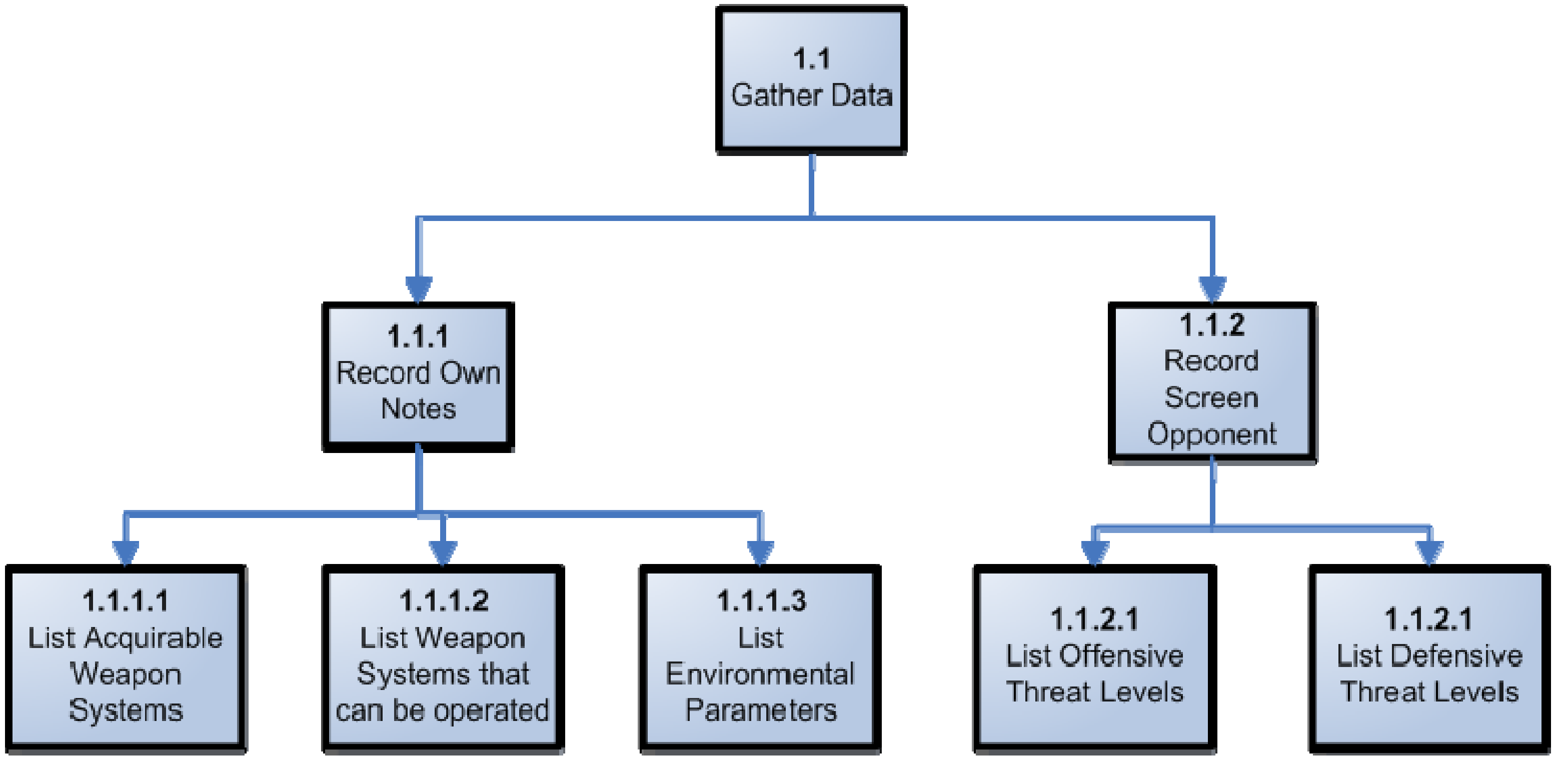

1.1
Gather Data
1.1.1
Record Own Notes
1.1.2
Record Screen Opponent
1.1.1.1
List Acquirable Weapon Systems
1.1.1.2
List Weapon Systems that can be operated
1.1.1.3
List Environmental Parameters
1.1.2.1
List Offensive Threat Levels
1.1.2.1
List Defensive Threat Levels

Figure 31. Functional Decomposition for 'Gather Data' Function

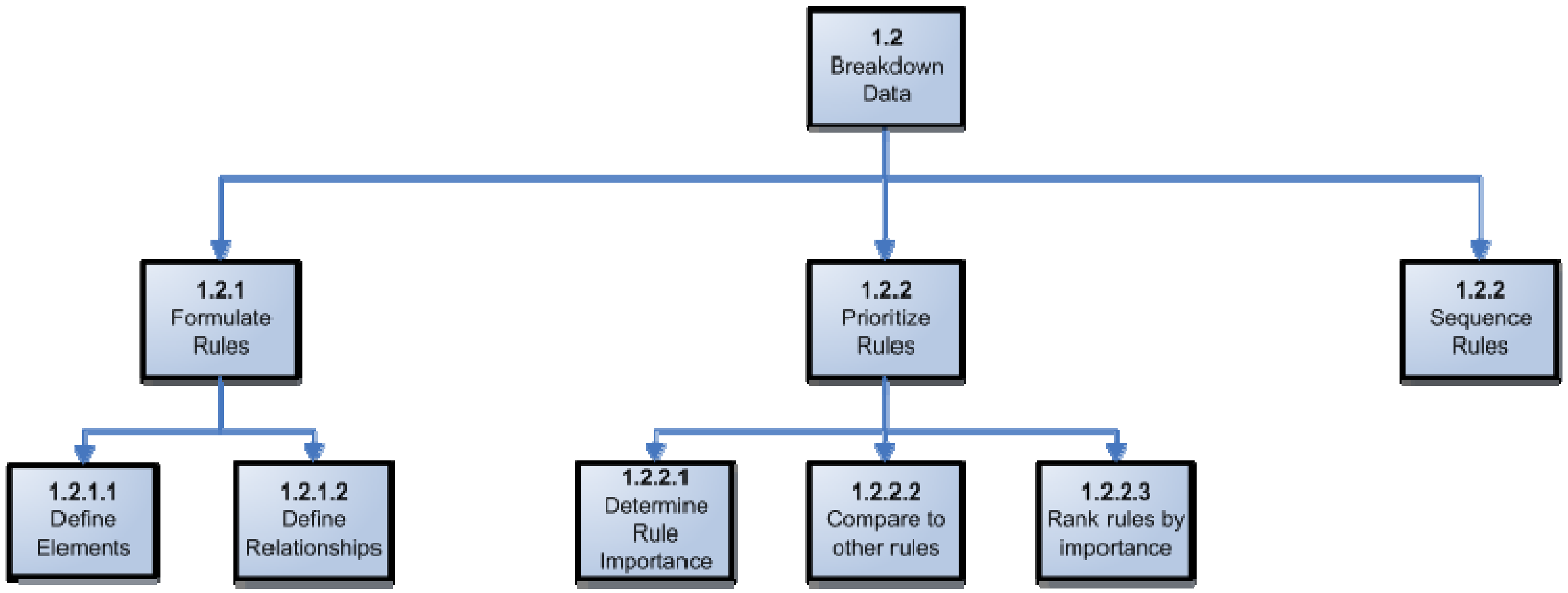


Figure 32. Functional Decomposition for 'Breakdown Data' Function

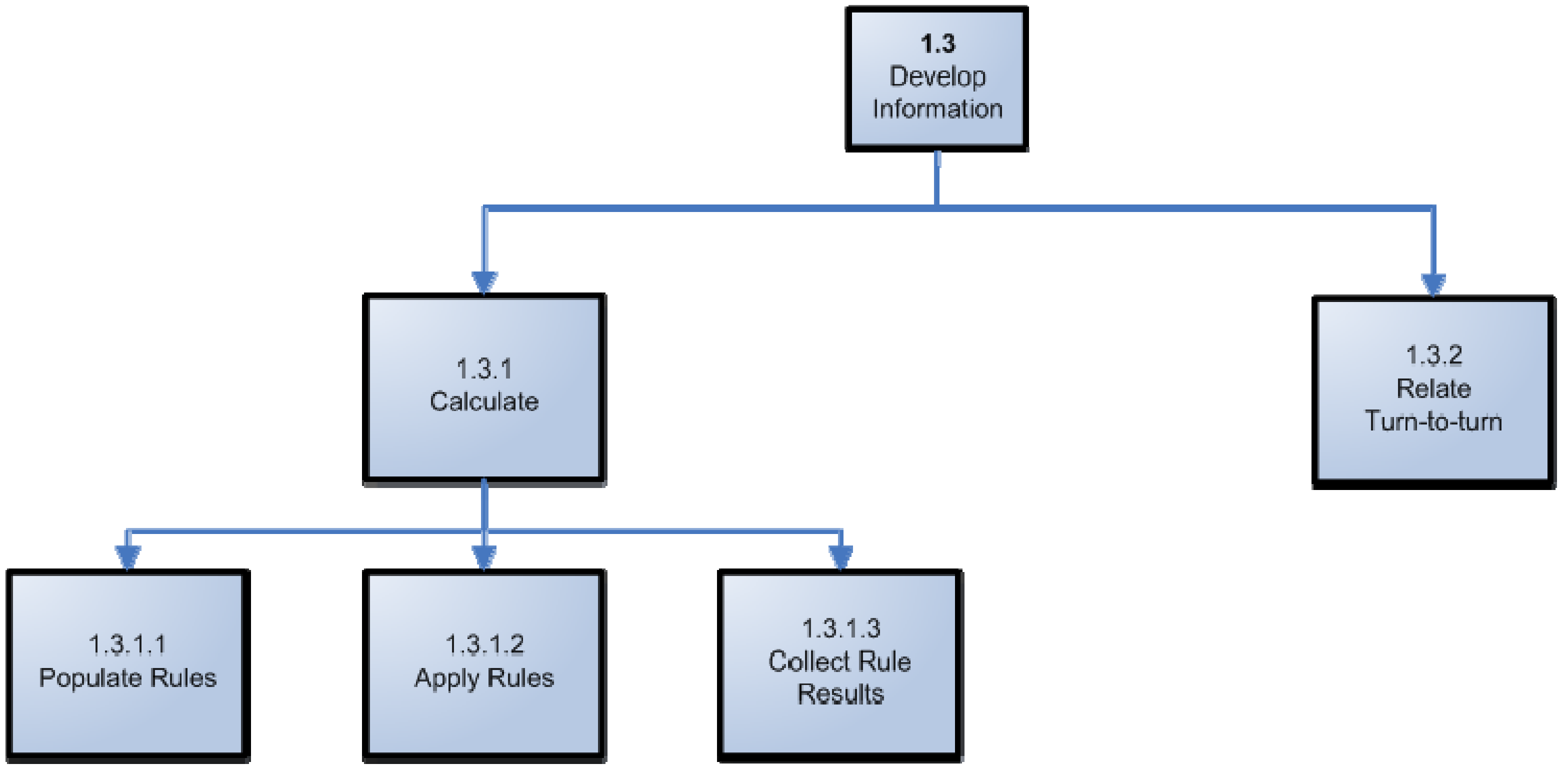


Figure 33. Functional Decomposition for 'Develop Information' Function

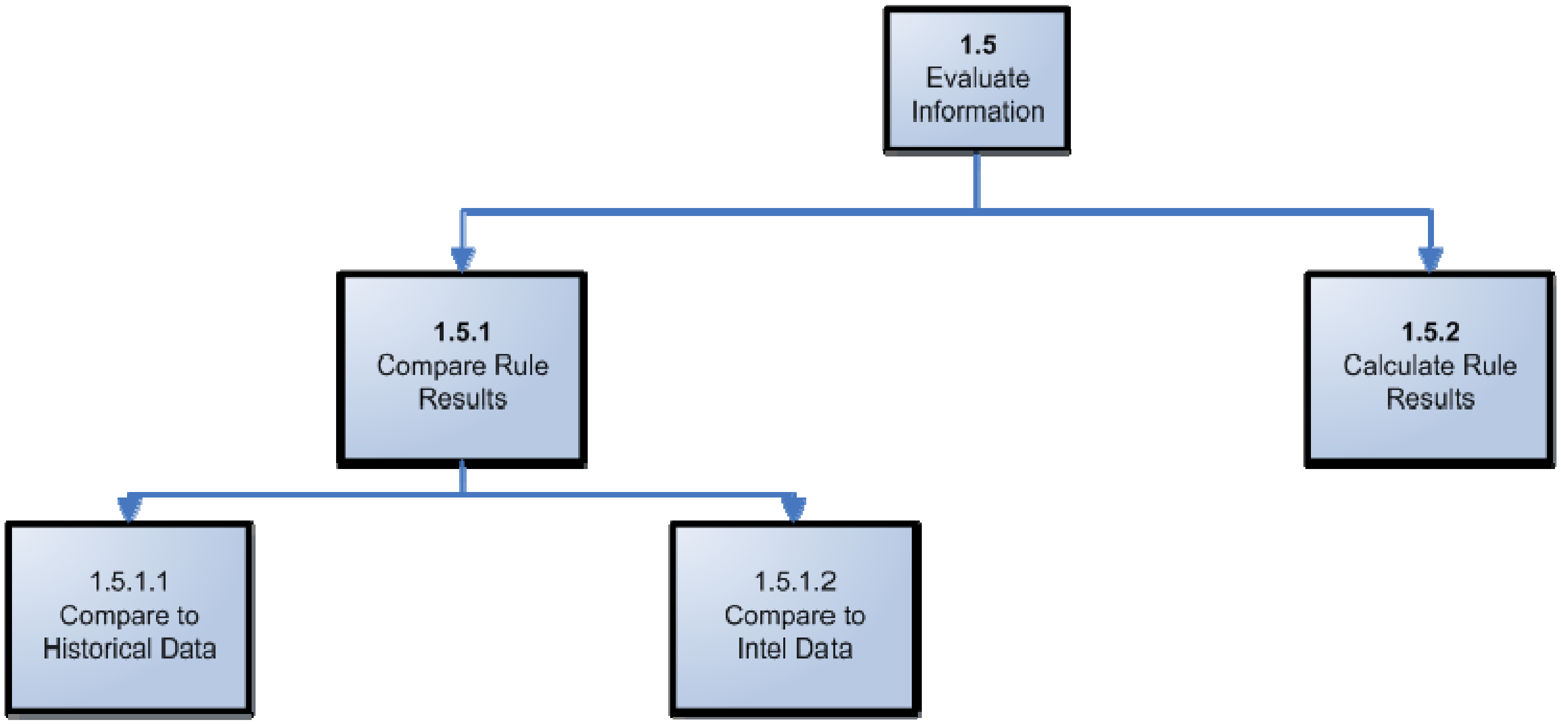


Figure 34. Functional Decomposition for "Evaluate Information" Function

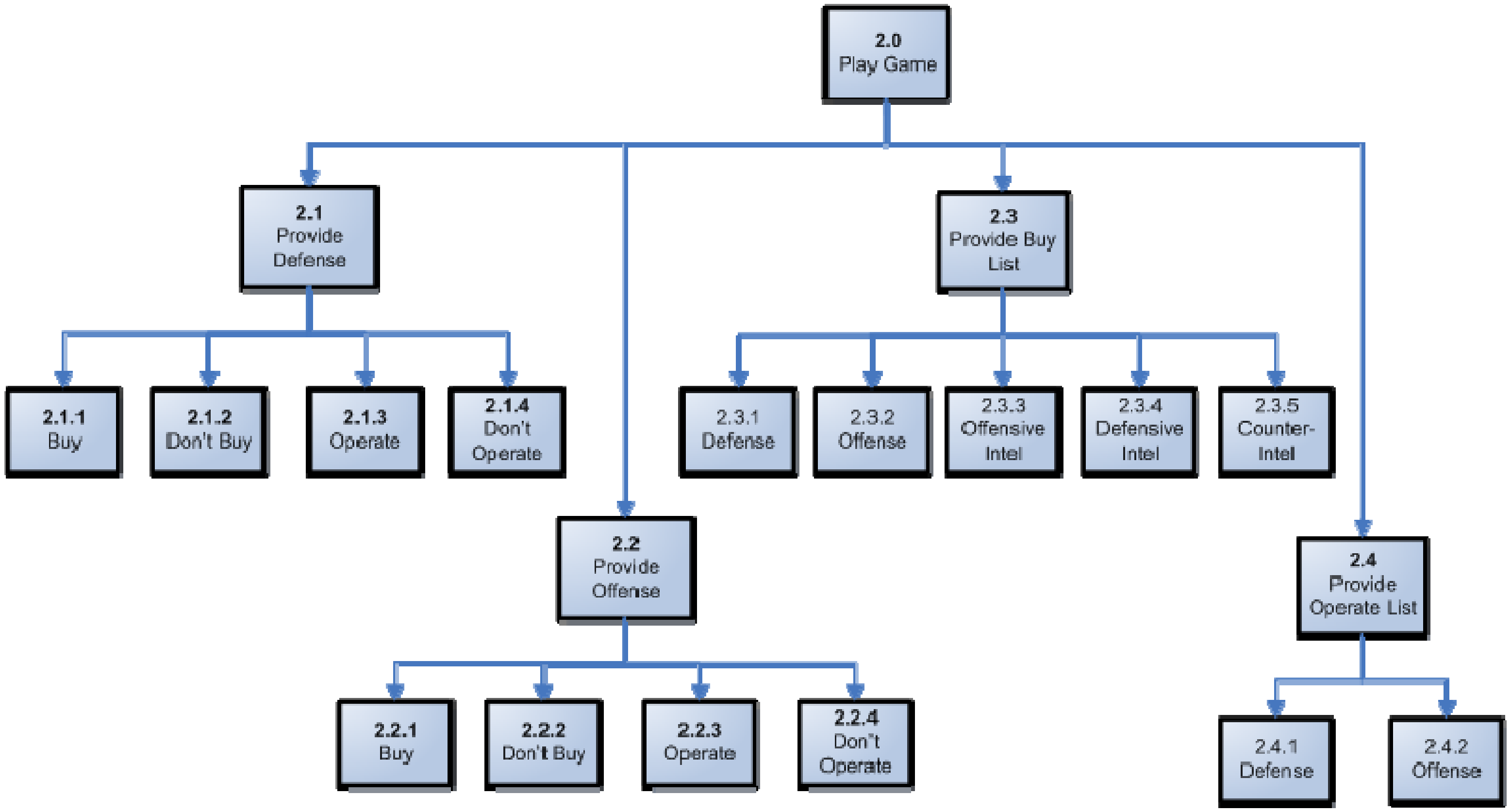


Figure 35. Functional Decomposition for "Play Game" Function

Table 2 describes function definitions. Only lowest level functions are defined. Higher level functions are defined by implementing their respective lower-level functions.

Table 2. Function Descriptions

Function ID	Description
1.0 Analyze Situation	
1.1 Gather Data	
1.1.1 Record Own Notes	
1.1.1.1 List Acquirable Weapon Systems	The act of recording, in some physical manner, all available weapon systems that can be bought for turn n
1.1.1.2 List Weapon Systems that can be Operated	The act of recording, in some physical manner, all weapon systems, and the quantities available, that are currently being stored in inventory to operate in turn n+1
1.1.1.3 List Environmental Parameters	The act of recording, in some physical manner, probability of war and budget for turn n
1.1.2 Record Screen Opponent	
1.1.2.1 List Offensive Threat Levels	If offensive intel is purchased during turn n-1, the act of recording offensive intel results for net utils
1.1.2.2 List Defensive Threat Levels	If defensive intel is purchased during turn n-1, the act of recording defensive intel results for net utils
1.2 Breakdown Data	
1.2.1 Formulate Rules	
1.2.1.1 Define Elements	List all possible variables within the TEMPO model that can be represented
1.2.1.2 Define Relationships	List constraining interconnections between variables (i.e., functions/equations using above defined variables)
1.2.2 Prioritize Rules	
1.2.2.1 Determine Rule Importance	Evaluate possible effects of rule on model outputs before each turn and calculate relative weights using pairwise comparison method
1.2.2.2 Compare to Other Rules	Ensure complete set of rules exist
1.2.2.3 Rank Rules by Importance	Collect all rules together with weightings and sort in order of rank
1.2.3 Sequence Rules	Arrange rules in the order that allows the specific algorithm to be implemented
1.3 Develop Information	
1.3.1 Calculate	
1.3.1.1 Populate Rules	Insert applicable variables into given rules
1.3.1.2 Apply Rules	Determine rule outputs with given variable inputs
1.3.1.3 Collect Rule Results	Gather rule outputs to prepare for evaluation and information generation
1.3.2 Relate Turn-to-turn	

Function ID	Description
1.4 Collect Historical Game Data	Gather data from environmental and performance files from previous game outcomes and collate into organized inputs for comparison
1.5 Evaluate Information	
1.5.1 Compare Rule Results	
1.5.1.1 Compare to Historical Data	If Historical game data is collected (Function 1.4), judge performance of each rule against historical data
1.5.1.2 Compare to Intel Data	If Offensive or Defensive Intel (Function 2.3.3 or 2.3.4) is purchased, judge performance of each rule against corresponding Intel data
1.5.2 Calculate Algorithm Results	Provide input for 2.3 based on outputs from 1.3.1.3
2.0 Play Game	
2.1 Provide Defense	
2.1.1 Buy	Select defensive weapon systems to buy based on output from Function 2.3.1.
2.1.2 Don't Buy	Bypass purchasing this defensive weapon system
2.1.3 Operate	Select defensive weapon system to operate based on output from Function 2.4.1.
2.1.4 Don't Operate	Bypass operating this defensive weapon system
2.2 Provide Offense	
2.2.1 Buy	Select offensive weapon systems to buy based on output from Function 2.3.2.
2.2.2 Don't Buy	Bypass purchasing this offensive weapon system
2.2.3 Operate	Select offensive weapon system to operate based on output from 2.4.2
2.2.4 Don't Operate	Bypass operating this offensive weapon system
2.3 Provide 'Buy' List	
2.3.1 Defense	Based on output from Function 1.5.2, list type and quantity of defensive weapon systems to acquire
2.3.2 Offense	Based on output from Function 1.5.2, list type and quantity of offensive weapon systems to acquire
2.3.3 Defensive Intel	Based on output from Function 1.5.2, display whether defensive Intel shall be purchased
2.3.4 Offensive Intel	Based on output from Function 1.5.2, display whether offensive Intel shall be purchased
2.3.5 Counter-Intel	Based on output from Function 1.5.2, display whether counter-intel shall be purchased
2.4 Provide 'Operate' List	
2.4.1 Defensive	Based on output from Function 1.5.2, list type and quantity of defensive weapon systems to operate
2.4.2 Offensive	Based on output from Function 1.5.2, list type and quantity of offensive weapon systems to operate

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX D. VERIFICATION MATRICES FOR EACH STRATEGY

Verification of each strategy was executed once the first version of each strategy was coded. It involved decomposing the TEMPO game into multiple worst-case scenarios and verifying the strategies worked under each. The log tables below display which strategies worked in which situations. Each strategy not using intelligence is verified by 16 figures—one for each extreme situation described in Table 3. Currently, only one strategy was implemented to use the Intelligence reported by TEMPO. This strategy is verified by 32 figures—one for each extreme situation described in Table 4. All figures are presented within the 'Strategy Verification' on the CD containing the entire TEMPO Version 3 software package to demonstrate that strategy verification was performed.

Table 3. Tested Categories for all Strategies Except 'Intel'

Weapons available	Max Budget Available	No Budget Available
All	☑	☑
None	☑	☑
OA	☑	☑
OB	☑	☑
Offensive	☑	☑
DA	☑	☑
DB	☑	☑
Defensive	☑	☑

Table 4. Tested Scenarios for 'Intel' Strategy

Weapons available	**Max Budget Available**		**No Budget Available**	
	Enemy Purchased…	**No Intel**	**Enemy Purchased…**	**No Intel**
All	☑	☑	☑	☑
None	☑	☑	☑	☑
OA	☑	☑	☑	☑
OB	☑	☑	☑	☑
Offensive	☑	☑	☑	☑
DA	☑	☑	☑	☑
DB	☑	☑	☑	☑
Defensive	☑	☑	☑	☑

APPENDIX E. SOFTWARE PACKAGE INSTALLATION INSTRUCTIONS

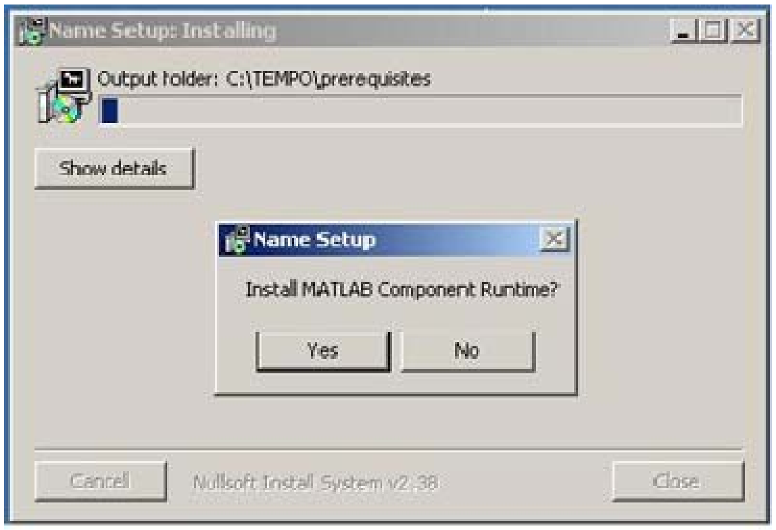
For first-time installation:

1. Download file and rename from: 'TEMPO_Setup.exx' to: 'TEMPO_Setup.exe'. This file is 143 MB in size, so it could take some time depending on your internet connection speed.

   a. Double-click 'TEMPO_Setup.exe'

   b. Select 'Yes' to install the MATLAB Component Run-time (see Figure 36).

NOTE: This installation will take from 2 minutes to 10 minutes, depending on computer.


Name Setup: Installing

Output folder: C:\TEMPO\prerequisites

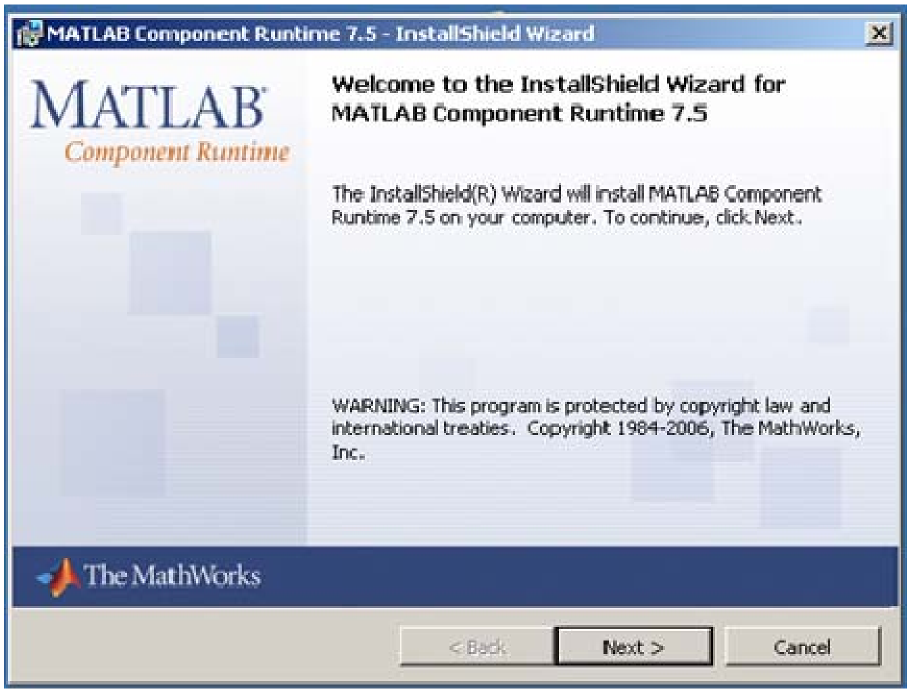
Name Setup

Install MATLAB Component Runtime?

Nullsoft Install System v2.38

Figure 36. Install MATLAB Component Runtime (MCR) Screenshot

1. The MCR installer will extract (MCRInstaller.exe). You will then be prompted to select a language. Continue to install the MATLAB Component Runtime (see Figure 37).


MATLAB Component Runtime 7.5 - InstallShield Wizard

MATLAB
Component Runtime

Welcome to the InstallShield Wizard for MATLAB Component Runtime 7.5

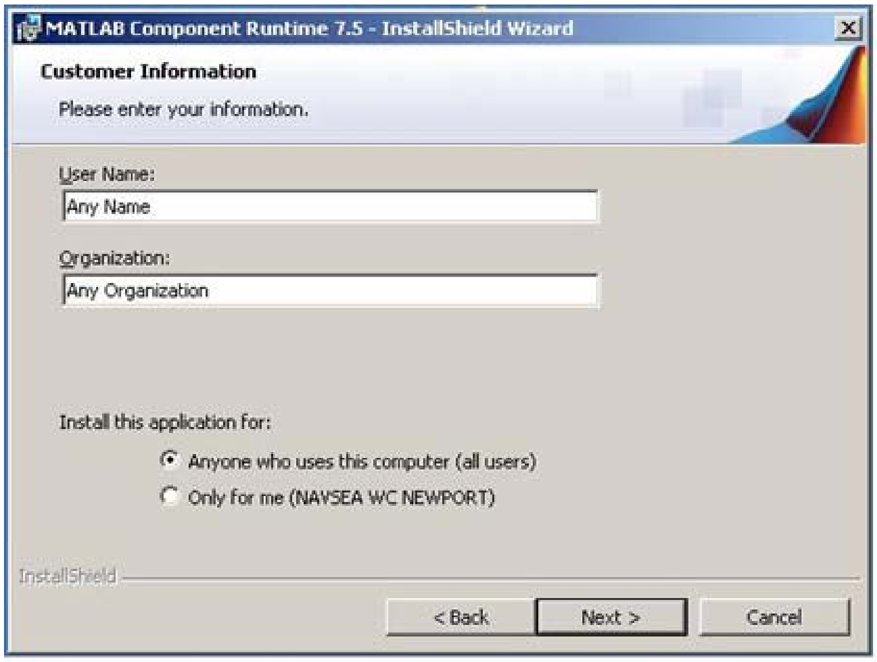
The InstallShield(R) Wizard will install MATLAB Component Runtime 7.5 on your computer. To continue, click Next.

WARNING: This program is protected by copyright law and international treaties. Copyright 1984-2006, The MathWorks, Inc.

The MathWorks

Figure 37. MATLAB Component Runtime Installation Screenshot

2. Enter appropriate name and organization for your computer. This is irrelevant to TEMPO operations (see Figure 38).


MATLAB Component Runtime 7.5 - InstallShield Wizard

Customer Information

Please enter your information.

User Name: Any Name

Organization: Any Organization

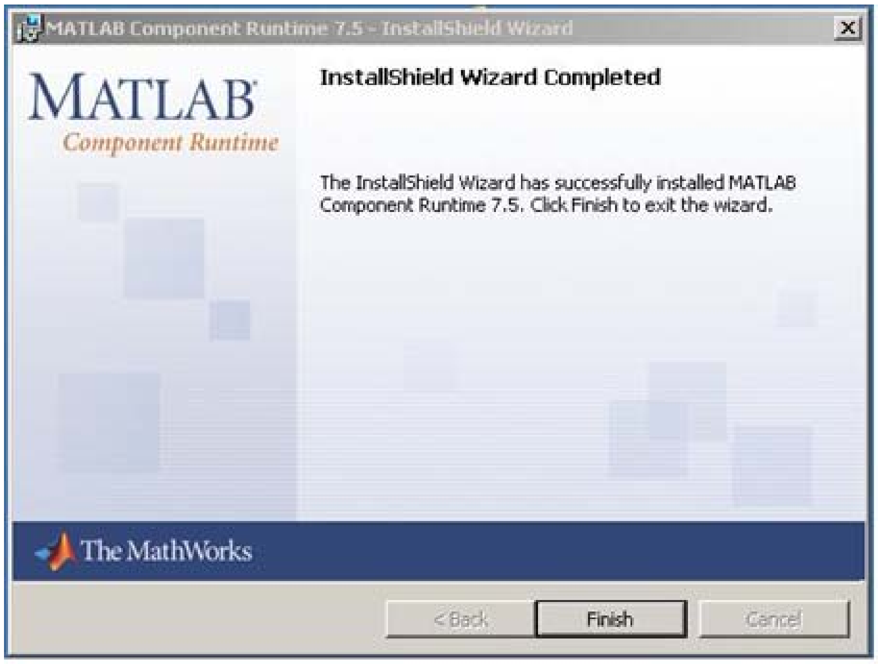
Install this application for:

[✓ Anyone who uses this computer (all users)]

Only for me (NAVSEA WC NEWPORT)

Figure 38. MCR Installation Data Required Screenshot

3. Continue installation to the default directory (i.e., 'C:\Program Files\...') (Figure 39).

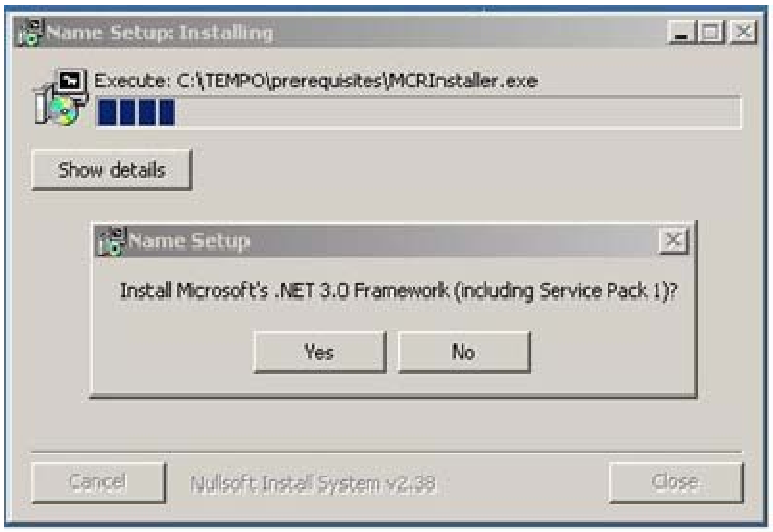

MATLAB Component Runtime 7.5 - InstallShield Wizard

InstallShield Wizard Completed

The InstallShield Wizard has successfully installed MATLAB Component Runtime 7.5. Click Finish to exit the wizard.

Figure 39. MCR Installation Complete Screenshot

a. Click 'Yes' to install Microsoft .NET Framework 3.0 Service Pack 1. Note: By clicking this link you will be directed to a Microsoft network location (see Figure 40).


Name Setup: Installing

Execute: C:\TEMPO\prerequisites\MCRInstaller.exe

Name Setup

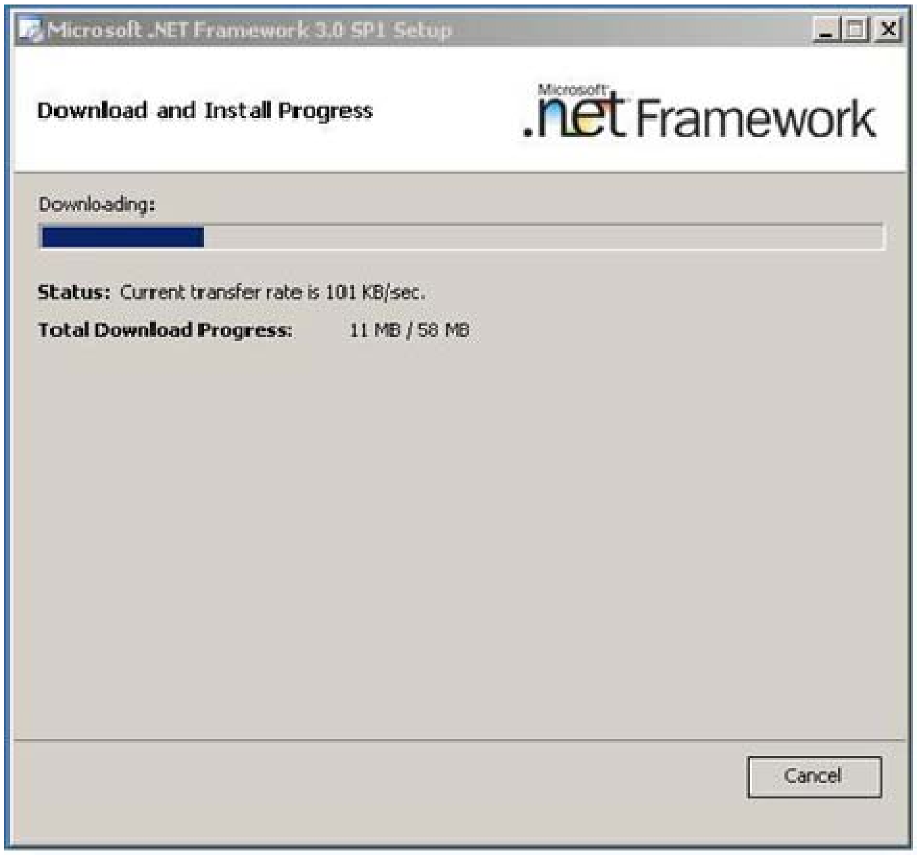
Install Microsoft's .NET 3.0 Framework (including Service Pack 1)?

Nullsoft Install System v2.38

Figure 40. Install Microsoft's .NET 3.0 Framework Screenshot

b. Accept the terms of the license agreement.

2. The installation package will then begin downloading. The total download size is 58 MB (see Figure 41).


Microsoft .NET Framework 3.0 SP1 Setup

Download and Install Progress

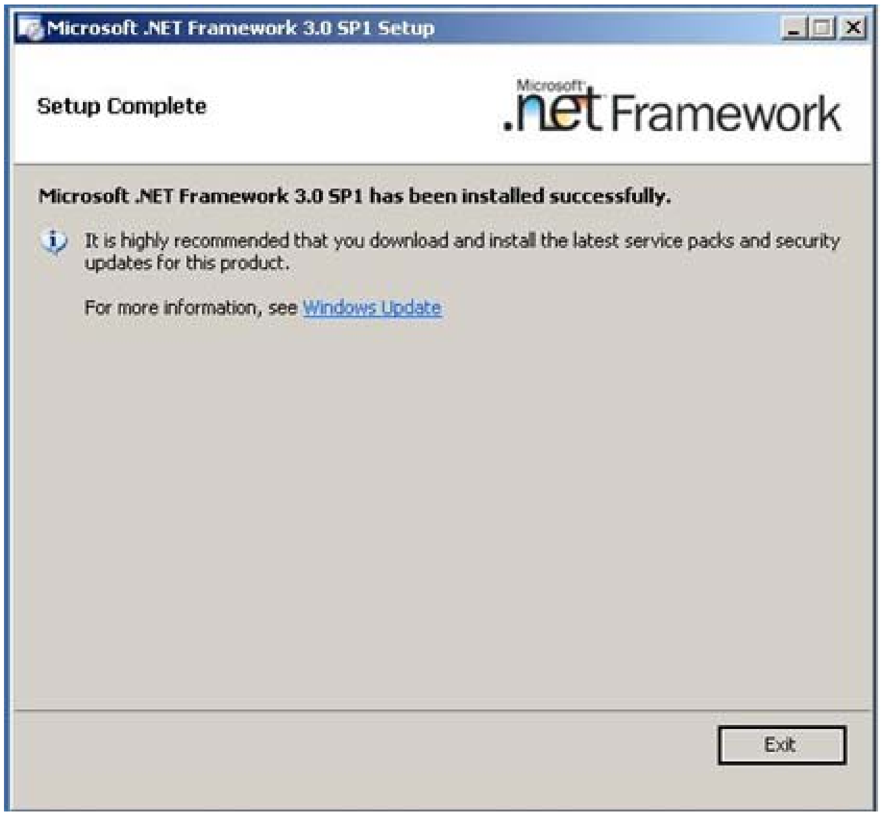
Downloading:

Status: Current transfer rate is 101 KB/sec.

Total Download Progress: 11 MB / 58 MB

Figure 41. Microsoft's .NET 3.0 Framework… Downloading... Screenshot

a. The installation will automatically occur and complete (see Figure 42). Visit 'Windows Update' if you desire to download the latest service packs and security updates.


Microsoft .NET Framework 3.0 SP1 Setup

Setup Complete

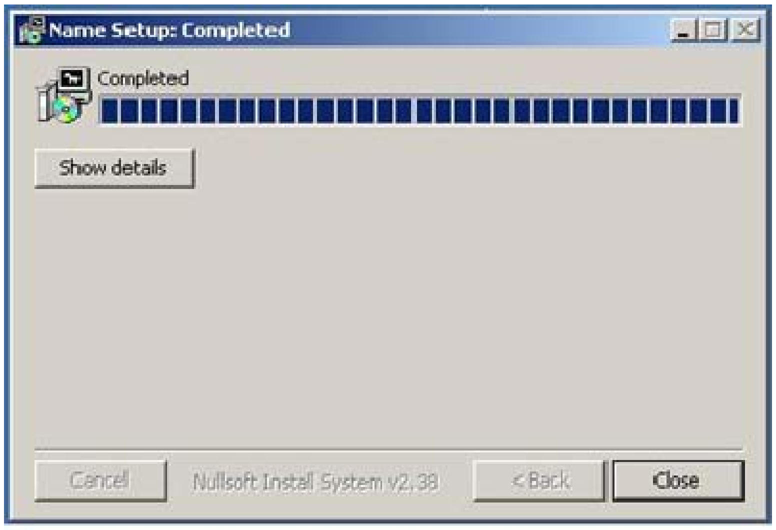
Microsoft .NET Framework 3.0 SP1 has been installed successfully.

It is highly recommended that you download and install the latest service packs and security updates for this product.

For more information, see Windows Update

Figure 42. Microsoft's .NET 3.0 Framework Installation Complete Screenshot

b. Click ‘Exit’ and the TEMPO.NET installation will continue and complete (see Figure 43).


Name Setup: Completed

Completed

Figure 43. TEMPO.NET Installation Complete Screenshot

To run/play TEMPO:

1. To run TEMPO.NET navigate to 'Start' → 'All Programs' → 'NPS TEMPO' (see Figure 44).

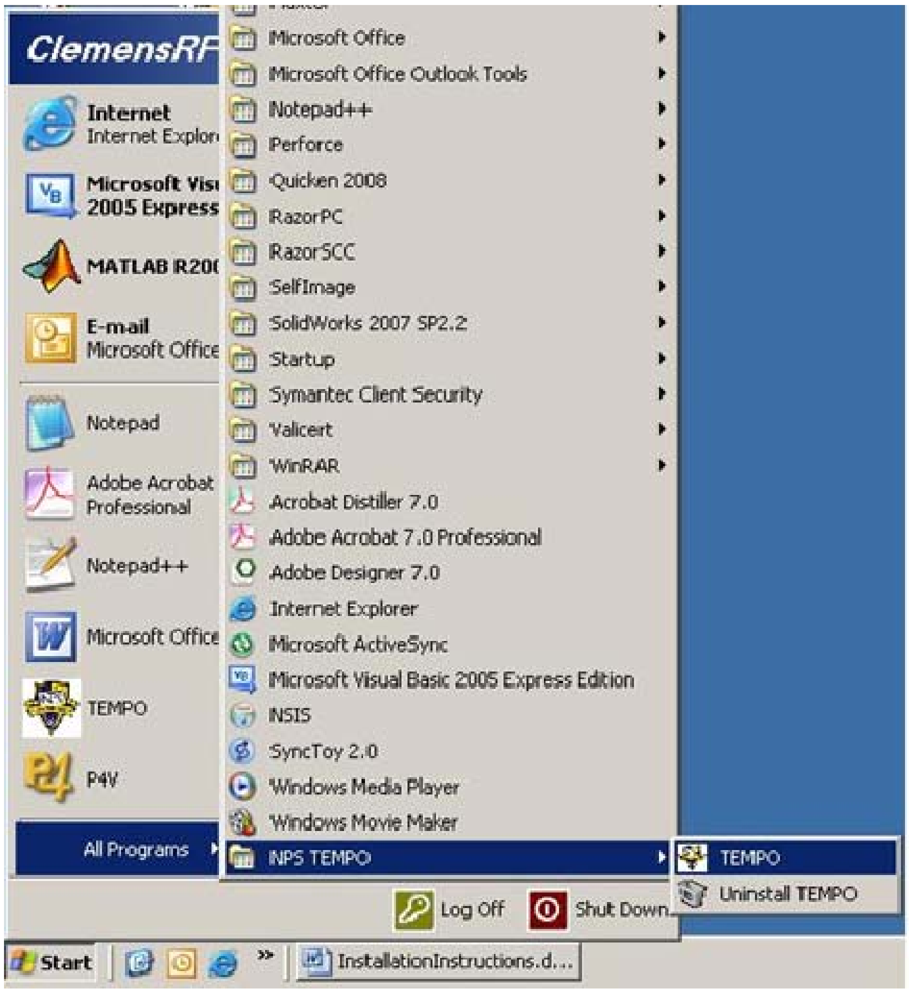
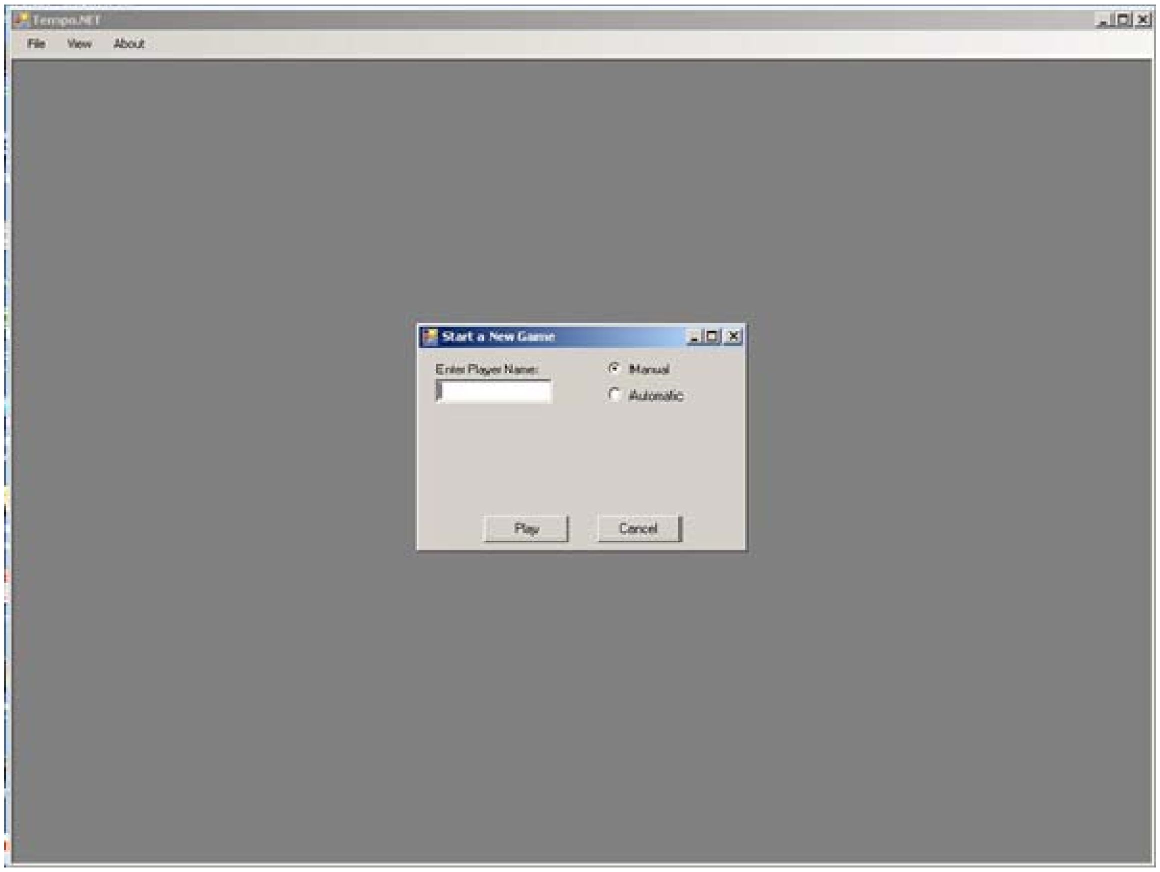


Figure 44. Run TEMPO Screenshot

2. The 'Start a New Game' Window will appear (see Figure 45). If at any point this window is not visible, click on 'File' → 'New Game'. If you are unable to do so, the window is hidden. To unhide, while focused on the TEMPO.NET Application, hold the 'Alt' key and continue to press 'Tab' until the icon is highlighted. When released, the 'Start a New Game' window will reappear.



Enter Player Name:

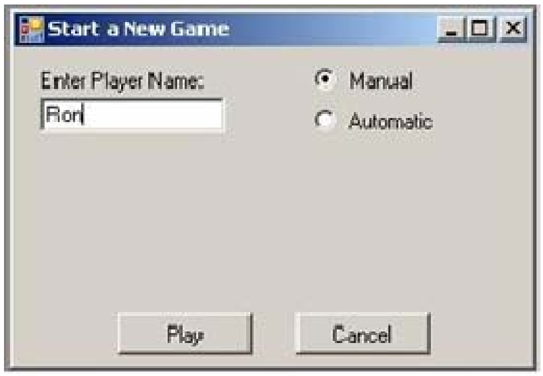
Manual [✓]

Automatic

Play Cancel

Figure 45. Game Startup Screenshot

3. At this point, the choice to play a 'Manual' Game (human vs. computer) or an 'Automatic' Game (strategy vs. computer) is presented. Both screens are shown below in Figure 46.

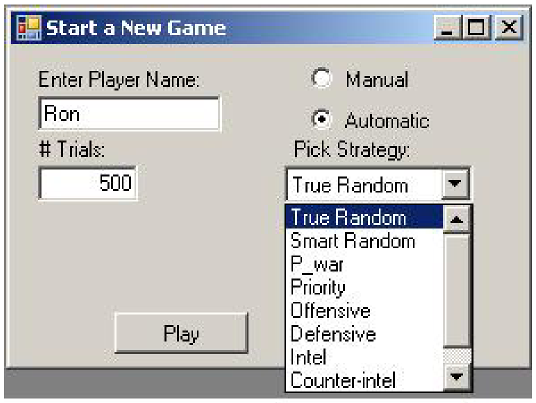


Enter Player Name: Ron

Manual [✓]

Automatic

Play Cancel



Enter Player Name: Ron

Manual

Automatic [✓]

Trials: 500

Pick Strategy: True Random

Figure 46. 'Start a New Game' Options Screenshots

4. For a ‘Manual’ game, skip to step 5. For an ‘Automatic’ game, first enter the number of desired trials. Then, select a strategy. In this version, the user can select any strategy, however, only the first six strategies will work. Other strategies will cause the application to hang up. Please only choose one of the first 6 to use.

NOTE: Each trial, takes anywhere between 10 seconds and 2 minutes, depending on the number of turns executed before war occurs. Please consider this timing when choosing the number of trials!

5. Once a selection is made, click ‘Play’ to begin!

NOTE: Please be patient. It does take some time for the first run to initialize and execute after a fresh installation—sometimes as long as 2 minutes. Initialization of the MATLAB component runtime code causes this significant delay.

6. After play is completed, the associated log file can be viewed. To do so, navigate to ‘C:\TEMPO\logs’. All applicable log files are stored here, in XML format.

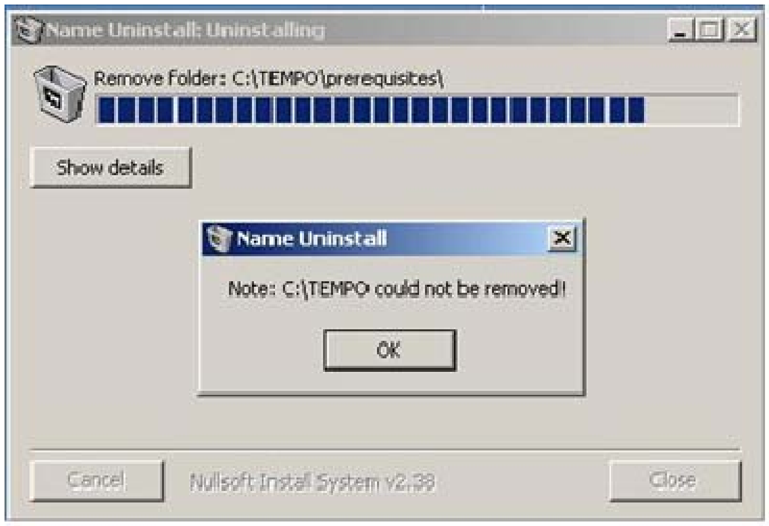
To uninstall TEMPO:

WARNING: The Uninstallation process will remove ALL TEMPO.NET log files located within the ‘C:\TEMPO’ directory. If this is undesired, move these files to a separate folder outside of the ‘C:\TEMPO’ directory structure!

1. There are two methods to uninstall TEMPO.NET. The first method is to navigate to ‘Start’ → ‘All Programs’ → ‘NPS TEMPO’ and click ‘Uninstall TEMPO’. The second method is to navigate to ‘C:\TEMPO’ and click the ‘Uninstall TEMPO’ icon. Both will initialize the uninstaller.

    a. The user should select whether to remove the MATLAB component runtime from the machine. Do this only if you do not plan to run TEMPO.NET again on the machine.

    b. If the TEMPO.NET application has not been played after the installation, the removal will be complete (i.e., all associated files and directories will automatically be removed). However, if TEMPO.NET has been executed at least once, the user must manually delete the ‘C:\TEMPO’ directory. This will be apparent if a message box appears during the removal process (see Figure 47).


Name Uninstall: Uninstalling

Remove folder: C:\TEMPO\prerequisites\

Name Uninstall

Note: C:\TEMPO could not be removed!

Figure 47. TEMPO.NET Directory Removal Screenshot

c. After deleting 'C:\TEMPO' the TEMPO.NET application removal is complete!

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

Abraham, A. Jain, L. C., & Goldberg, R. (2005). Evolutionary multiobjective optimization: Theoretical advances and applications. London: Springer-Verlag.

Al Mannai, W. I. & Lewis, T.G. (2007). Minimizing network risk with application to critical infrastructure protection. *Journal of Information Warfare, 6*(2), 52–68.

Baeck, T., Fogel, D.B., & Z. Michalewicz, Z. (2000).*Evolutionary computation: basic algorithms and operators.* Boca Raton, FL: CRC Press.

Bernoulli, D. (1954). Exposition of a new theory on the measurement of risk. (L. Sommer, Trans). *Econometrica, 22*(1), 22–36. (doi:10.2307/1909829.) (Original work published 1738). Retrieved May 30, 2006, from http://www.math.fau.edu/richman/Ideas/daniel.htm

Chapman, G. B. & Johnson, E. J. (1995). Preference reversals in monetary and life expectancy evaluations. *Organizational Behavior and Human Decision Processes, 62*, 300–317.

DoD Handbook 4245.8. "Value Engineering." (1986). Authorized by DoD Directive 4245.8. Arlington, VA: Defense Technical Information Center.

Doob, J.L. (1971). What is a martingale? *The American Mathematical Monthly, 78*(5), 461–473.

Federal Aviation Administration. (2006). Trade studies section 4.6 in *System Engineering Manual Version 3.1*. Retrieved May 30, 2006, from http://www.faa.gov/about/office_org/headquarters_offices/ato/service_units/operations/sysengsaf/seman/SEM3.1/Ch_1.pdf

Field, K.A. (1999, July). Fast track to management. *Design News*.

Game Theory. Pareto efficient. Retrieved May 30, 2006, from http://www.gametheory.net/dictionary/ParetoEfficient.html

Howard, R. A. (1960). *Dynamic programming and Markov processes*. Cambridge, MA: MIT Press.

Hubertus, T. J., Meer, K., & Triesch, E. (2004). *Optimization theory*. New York: Springer.

Johnson, R.W., et al. (2005). Coevolutionary optimization of fuzzy logic intelligence for strategic decision support. *IEEE Transactions on Evolutionary Computation, 9*(6), 682–694.

Keeney, R. L., & Raiffa, H. (1976). *Decisions with multiple objectives*. Cambridge, England: Cambridge University Press.

Krantz, D. H., and Kunreuther, H.C. (2007). Goals and plans in decision making. *Judgment and Decision Making,* 2 (3), 137–168.

Lakatos, I. (1978). *The Methodology of Scientific Research Programmes: Volume 1: Philosophical Papers.* Cambridge, England: Cambridge University Press.

Langford, G., & Huynh, T. (2007, September). *A methodology for managing complexity, systems engineering test and evaluation.* Presented at Complex Systems and Sustainability Conference, Sydney, Australia.

Langford, G.O. (2007). Reducing risk in designing new products using rapid Systems Engineering Proceedings, *Asia-Pacific Systems Engineering Conference, paper no. 18*. Singapore.

Langford, G.O. (2006). Reducing risk of new business start-ups using rapid Systems Engineering. *Proceedings of the Fourth Annual conference on Systems Engineering Research, paper no. 140, Los Angeles, CA.*

Langford, G.O., Franck, R., Huynh, T., & Lewis, I., (2007). Gap analysis: rethinking the conceptual foundations. (Report No. NPS-AM-07-051). Monterey, CA: Naval Postgraduate School.

Langford, G. O. (2007, March). *Reducing Risk in Designing New Products Using Rapid Systems Engineering*. Paper presented at Asia-Pacific Systems Engineering Conference, Singapore.

Lewis, T. (2006). *Critical infrastructure protection in homeland security*. Hoboken, NJ: John Wiley & Sons.

Lowrance, W. W. (1976). *Of acceptable risk*. Los Altos, CA: William Kaufman, Inc.

Management Concepts. (2003). *Defense acquisition guidebook*. (Republished in 2008). Vienna, VA: Management Concepts Inc.

Miles, L. D. (1972). *Techniques for value analysis and engineering* (2nd ed.). New York: McGraw Hill.

Resnik, M. (1987). *Choices: An introduction to decision theory*. Minneapolis, MN: University of Minnesota Press.

Saaty, T.L. (2001). *The analytic network process: decision making with dependence and feedback*. Pittsburgh, PA: RWS Publications.

Saaty, T.L. (2001). *Decision making for leaders: the analytic hierarchy process for decisions in a complex world*. Pittsburgh, PA: RWS Publications.

Slovic, P. (1995). The construction of preference. *American Psychologist*, *50*, 364–371.

Steuer, R.E. (1986). *Multiple criteria optimization: Theory, computations, and application.* New York: John Wiley & Sons, Inc.

Sutton, R. S. & Barto, A. G. (1998). *Reinforcement learning: An introduction*. Cambridge, MA: MIT Press.

Taguchi, G. & Chowdhury, S., & Wu, Y. (2005). *Taguchi's quality engineering handbook*. Hoboken, NJ: Wiley-Interscience.

Taguchi, G. (1990). *Introduction to quality engineering*. Tokyo, Japan: Asian Productivity Organization.

TEMPO military planning game, explanation and rules for players. (n.d.). Internal document, Naval Postgraduate School, CA.

Tversky, A., Sattath, S. & Slovic, P. (1988). Contingent weighting and judgment and choice. *Psychological Review*, *95*, 371–384.

Tversky, A., Slovic, P. & Kahneman, D. (1988). The causes of preference reversal. *American Economic Review, 80*, 204–217.

Zadeh, L. (1965). Fuzzy Sets. *Information and Control*, *8,* 338–353.

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST